IBM ILOG OPL V6.3

IBM ILOG OPL IDE Reference

Copyright

Trademarks

Acknowledgement

The language manuals are based on, and include substantial material from, The OPL Optimization Programming Language by Pascal Van Hentenryck, © 1999 Massachusetts Institute of Technology.

Table of contents

IDE Reference....7
About the IDE GUI Reference Manual....8
Tour of the OPL Graphical User Interface....9
The main window....11
General description....13
The Welcome window....15
The OPL Projects Navigator....18
Copying projects in OPL Projects Navigator....21
File types....22
Searching files....23
The Editing Area....25
The Outline....26
The Problem browser....28
The Status Bar....35
The Output Area....37
Toolbars....39
General description....40
The standard toolbar....41
The execution toolbar....43
The Debug views toolbars....47
The OPL Projects Navigator view toolbar....50
The Problem browser view toolbar....51
The Output Area tab views toolbars....52

The text editor....55
Editor features....57
Switching between Editor windows....58
Resizing an Editor window....59
Keyboard shortcuts....60
Model Completion and Model Templates....63
Local History and its related features....81
'Compare With' features....82
'Replace With' features....84
Menu commands and equivalent toolbar buttons....87
General description....88
File menu....89
Edit menu....91
Navigate menu....93
Run menu....94
Window menu....97
Help menu....98
Right-click context menu commands....99
For projects, models, and data....100
For run configurations....101
In the Output Area....102
Preferences and options....103
Setting IDE preferences....104
Setting programming options....105
Icons....107
Types in the Problem browser....108
Types and sets in call stacks....109
Processes and Procedures....111
What happens when you execute a run configuration....112
Doing more with the Problem Browser....115
Introduction....116
Browsing a model without execution....117
Element usage in browsing vs. solving....120
Navigating the model file....122
Postprocessing....124
Displaying individual table views....125
Resulting views....126
Doing more with projects....127
Reusing existing files and projects....128
Working with several projects....130
Using templates....131
Order of files....132
Saving a project....135

Closing projects....136
Generating output files....137
Generating a compiled model....138
Exporting external data....140
Exporting internal data....141
Launching OPL run configurations in the background....143
Running your projects in the background....144
Creating an oplrun launch configuration interactively....145
Creating a customized oplrun launch configuration....148
External Tools Dialog options....150
IDE Preferences....157
The Preferences dialog box....158
OPL IDE advanced settings....163
OPL IDE memory allocation....164
Index....165

IDE Reference

Provides reference information on the OPL graphical user interface.

In this section

About the IDE GUI Reference Manual
Provides overview information about this manual and its contents.

Tour of the OPL Graphical User Interface
Describes the IDE: main window, toolbars, text editor, menu commands and equivalent toolbar buttons, right-click context menu commands, preferences and options, and icons.

Processes and Procedures
Provides more detail on the execution process, on the Problem Browser, and on project management.

IDE Preferences
Describes how to customize the appearance and features of the graphical user interface and the behavior of the editor and other tools such as CVS by setting preferences for the IDE.

OPL IDE advanced settings
Describes command-line settings accepted by the OPL IDE.

OPL IDE memory allocation
Describes the oplide.ini file that controls memory allocation for the IDE.

About the IDE GUI Reference Manual

This manual provides reference information on the OPL graphical user interface, including:

- descriptions of the windows, menus, toolbars, buttons, etc.
- descriptions of common processes and procedures
- explanations of how to set preferences and options

Make sure you read How to read the OPL documentation for details of prerequisites, conventions, documentation formats, and other general information.

Tour of the OPL Graphical User Interface

Describes the IDE: main window, toolbars, text editor, menu commands and equivalent toolbar buttons, right-click context menu commands, preferences and options, and icons.

In this section

The main window
After a general description, describes the Start window, the OPL Projects Navigator, file types, the Editing Area, the Outline, the Problem browser, the Status Bar, the Output Area, and other features.

Toolbars
Describes the standard toolbar, the execution toolbar, and toolbars on individual views.

The text editor
Describes the editing features available in the Editing Area, explains how to switch between Editor windows and how to resize an Editor window, and provides keyboard shortcuts.

Menu commands and equivalent toolbar buttons
Lists the available OPL commands menu by menu.

Right-click context menu commands
Describes the OPL commands available on context menus when you right-click a project, model, data, settings file, or a run configuration, and when you right-click in the Problem browser or in the Output Area.

Preferences and options
A general presentation of the Preferences dialog box and of the settings editor for OPL, CP, and MP options.

Icons

Describes the icons used in the IDE to represent types in the Problem browser, and types and sets in call stacks.

The main window

After a general description, describes the Start window, the OPL Projects Navigator, file types, the Editing Area, the Outline, the Problem browser, the Status Bar, the Output Area, and other features.

In this section

General description
Presents an overview of the main window of the OPL IDE and its primary features and controls.

The Welcome window
Describes the OPL Welcome screen, how to access its links, and how to close it and begin working with OPL.

The OPL Projects Navigator
Describes the window in the IDE used to manage your projects by creating, adding, or removing resources.

Copying projects in OPL Projects Navigator
This section gives the procedure for copying OPL projects.

File types
Lists the different types of files used by an OPL project, along with their extensions.

Searching files
How to search in one or more files.

The Editing Area
Describes the Editing Area of the IDE and the different contents that may be displayed in it, depending on what you are doing at the time.

The Outline
Describes the window of the IDE that displays a project's resources in a hierarchical tree structure.

The Problem browser
Describes the window in the IDE that displays the structure of the run configuration last executed.

The Status Bar
Describes the area that displays messages about the current execution status of the IDE and information about files being edited.

The Output Area
Describes the area of the IDE where messages, error information, solutions, and results are displayed.

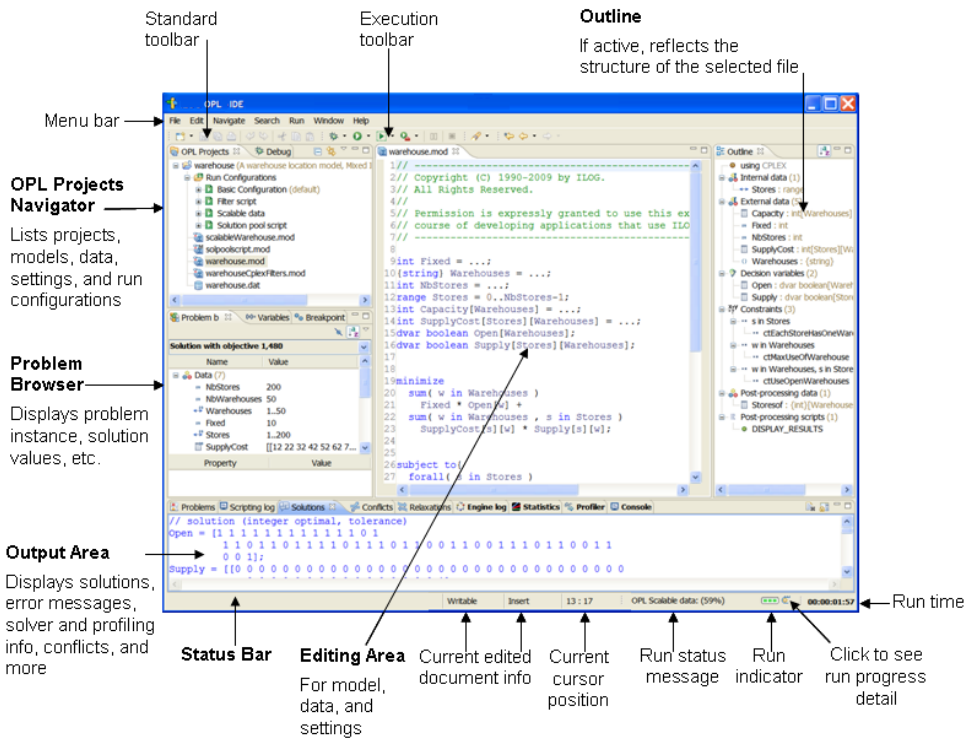
General description

When you launch the IBM® ILOG® OPL IDE (or IDE for short), the main window appears. All tasks and commands for using the IDE are carried out from this window.

The main window is divided into several dockable subwindows or *views*, which you can:

- resize within the main window using the sliding separators,
- expand to fill the whole IDE frame by double-clicking the tab for that view (double-click again to close it),
- move to other locations within the IDE frame by dragging them or
- detach and move to a location outside the IDE frame.

The Main window illustration below shows the main window with a project open.



The Main window

The menu bar is described in *Menu commands and equivalent toolbar buttons*.

Toolbars are described in *Toolbars*.

The other parts of the main window are described in more detail in:

- *The Welcome window*
- *The OPL Projects Navigator*
- *File types*
- *The Editing Area*
- *The Outline*
- *The Problem browser*
- *The Status Bar*
- *The Output Area*

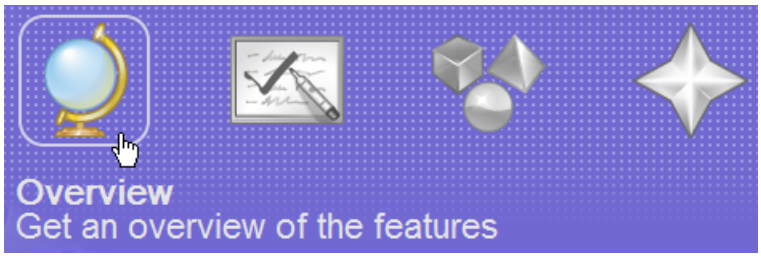
The Welcome window

When you first launch the OPL IDE, a Welcome window displays:

The Welcome window presents access to what's new in the release, to the Samples provided in the distribution, and to other information.

To access the information on the Welcome window:

- Mouseover the buttons to see a tooltip that explains them:



- Click the buttons to see the information.

The buttons contained on the Welcome window lead to information on the release and/or parts of the OPL documentation you might want to refer to often:

- **Overview** — this button displays an Overview page that contains:

- A link to the *From OR to OPL and ODM* section of the documentation.
- A link to the *Migrating from previous versions of OPL* section of the documentation.

♦ **Tutorials** — this button displays a page that contains a set of links to different sections of the *IDE Tutorials* manual.

♦ **Samples** — this button displays a page that contains a set of links to different sections of the Language and Interfaces Examples manual.

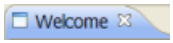
♦ **What's New** — this button displays an page that contains a set of links to:

- The *New and changed in OPL* section of the *Release Notes*.
- The *Introduction to the OPL IDE*, which provides an overview of features of the IDE.
- A set of links to various OPL and ODM user forums. These links are driven by an RSS feed, so they are constantly updated to reflect the latest information on those forums.

To close the Welcome window and use the OPL IDE:

♦ Click the Workbench icon at the top right of the Welcome window:

♦ **OR**, click the **X** in the Welcome window tab to close it.



Closing the Welcome window using the first method displays a Welcome window toolbar in the Status Bar, as shown below.

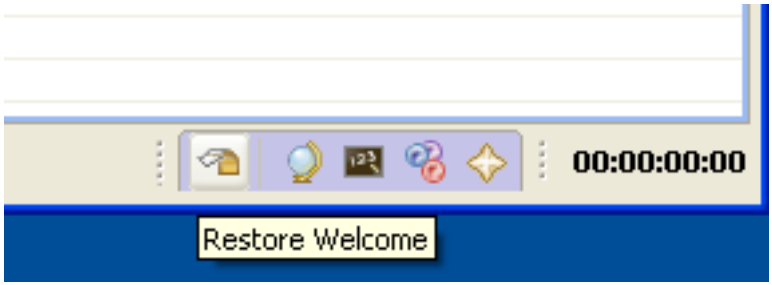
Closing the Welcome window by clicking the **X** in the tab does not display this toolbar.

♦ When you close the Welcome window using either method, the OPL Main window appears. It is described in the next section.

To return to the Welcome window from the OPL IDE:

♦ Choose **Help>Welcome** from the main menu.

♦ Or, click the **Restore Welcome** icon at the bottom right of the IDE. (It appears when you click the Workbench arrow.)

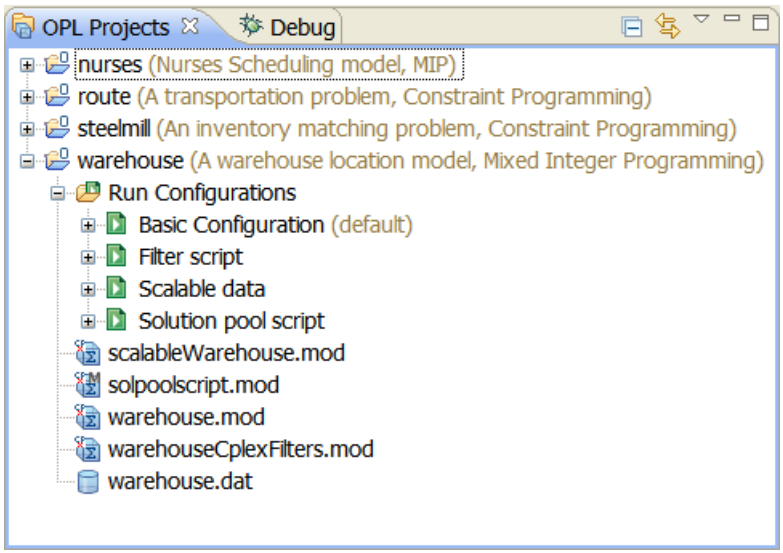

00:00:00:00
Restore Welcome

The OPL Projects Navigator

The OPL Projects Navigator is where you manage your projects by creating, adding, removing projects, models, data, settings, and run configurations. You can display more than one project at the same time (see *Working with several projects* in *IDE Reference*).

The OPL Projects Navigator displays projects as tree structures with the files below them listed in alphabetical order (except for within run configurations, where the order of files is important):

- the project name (and optional description) at the root
- an optional default run configuration (followed by 'default' in parentheses)
- one or more additional run configurations (optional)
- at least one model file
- data and settings files (optional)



OPL Projects Navigator

After you have migrated OPL 5.x projects, imported existing OPL 6.x projects, or created new OPL 6.x projects, you can leave them in your OPL Projects Navigator. When you next launch the OPL IDE, they will be there, ready to use.

If you have loaded a number of projects and the OPL Projects Navigator starts to get “crowded,” there are two ways to save space — by closing (collapsing) the projects or by deleting them.

Closing/Opening projects

Projects are either open or closed. When a project is closed, it cannot be changed, but its resources still reside on the local file system. Because they are not examined during builds,

closed projects require less memory. Therefore, closing projects you are not working with can improve build time.

- Right-click on the project name and choose **Close project** from the context menu to close the project.

  The plus sign next to the project name disappears, but it remains in the OPL Projects Navigator.

- To reopen the project, right-click on the project name and choose **Open project** from the context menu.

Deleting projects

If you are not currently working with a project, you can also safely delete it from the OPL Projects Navigator, without deleting it from the file system.

- To remove a project from the OPL Projects Navigator, right-click on the project name and choose **Delete** from the context menu.

  A popup message appears asking whether you want to delete the project from the file system or not.

- The two options available to you are:

  - Select **Also delete contents** to delete the project entirely.

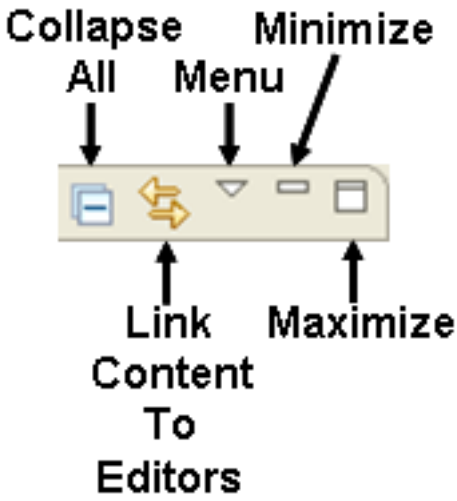
    This means that the project will be completely deleted, and cannot later be recovered using **Undo** or the **Import>Existing OPL 6.x projects** menu command.

  - Select **Do not delete contents** (the default) to remove the project from the OPL Projects Navigator but leave it on the file system.

    This means that the project is still present on the file system and can be reopened using the **Import>Existing OPL 6.x projects** menu command.

OPL Projects Navigator toolbar

The OPL Projects Navigator has a local toolbar. You can use its buttons to apply actions to projects, or to customize, minimize, or maximize the view. Mouseover each icon to see a tooltip that describes its function.


Collapse All
Menu
Minimize
Link Content To Editors
Maximize

Copying projects in OPL Projects Navigator

Often users want to make a copy of an existing OPL project to modify and test new features, while preserving the state of the existing project. You cannot do this easily by simply making a copy of the original project folder on the file system and then opening it in OPL, because even if you rename the model and data files in the copy folder, the copy will have the *same project name* as the original.

This means that you would not be able to have the original project and the copied project open at the same time, because you can't open two projects with the same project name. Therefore, the following procedure should be used if you want to copy an OPL project.

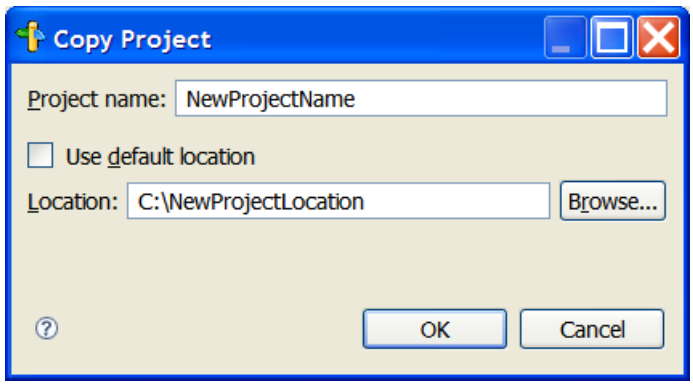
To copy a project in OPL Projects Navigator:

1. In OPL Projects Navigator, open the project you want to copy. Right-click on the project name and choose **Copy** from the context menu

   You could also select the project name and press **Ctrl + C**.

2. Right-click again and choose **Paste** from the context menu, or press **Ctrl + V**.

   The following popup window appears:



3. Change the project name to something other than the original project name or one of the other project names currently open in OPL, specify the location, and click **OK**.

   The new project appears in the OPL Projects Navigator.

File types

Each type of file in a project has a distinctive file name extension, as shown in *File extensions* .

File extensions

File Extension	Description
`.dat`	Files containing data instances.
`.mod`	Files containing modeling and scripting statements
`.opl`	Compiled model files
`.ops`	Settings files, containing user-defined values for OPL language options, CPLEX® parameters, and CP Optimizer parameters

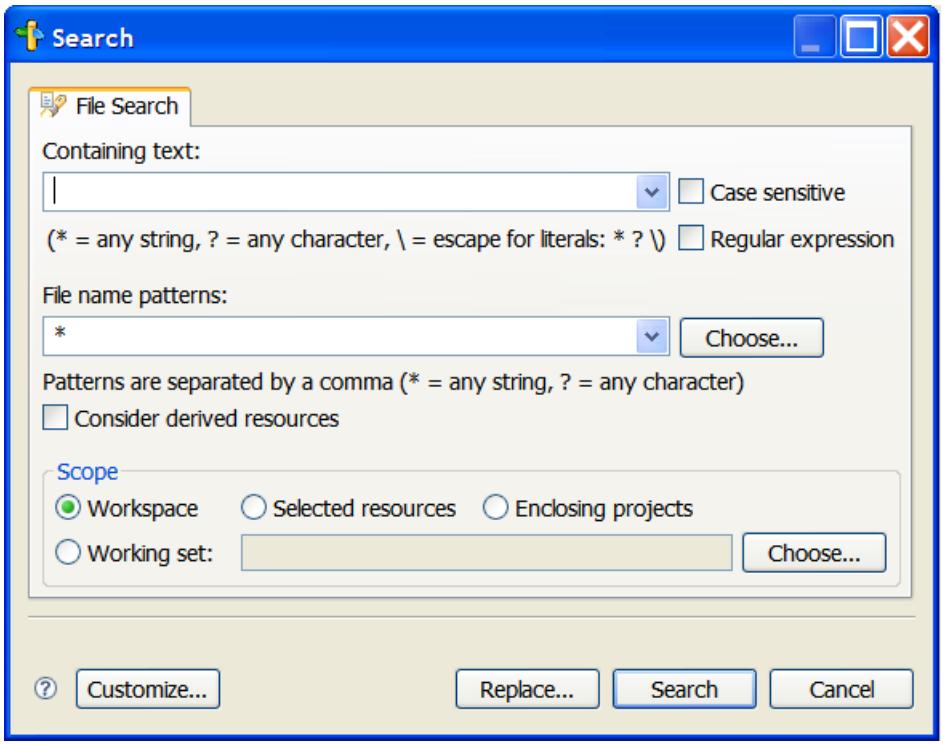
For more information, see:

- *Understanding OPL projects* in the *Quick Start* manual
- *Creating a project* in *Getting Started with the IDE*
- *Generating output files* in *IDE Reference*
- *The Problem browser view toolbar*
- *Right-click context menu commands*

Searching files

The OPL 6.x IDE incorporates more powerful search capabilities than were present in previous releases of OPL. You can search a single file or multiple files at once, and by a large number of criteria.

For example, when you click the **Search** toolbar button  or choose **Search** from the Search menu, you see the following window:



Using this window, you can search using the following criteria:

- **Case sensitivity** — you can turn case sensitivity on or off using the checkbox.
- **Containing text** — you can search for any text strings, using the following wildcards:
  - "*****" matches any set of characters, including the empty string.

    "**?**" matches any single character.

    "****" is the escape character for a literal; if you want to search for an asterisk, question mark, or backslash character, type a backslash before it to indicate that you are not using these characters as wildcards (for example, "*****", "**\?**", or "****")
  - You can also check the **Regular expression** box to indicate what you are searching for.

- **File name patterns** — you can search for one or more patterns, using the following wildcards:
  - "*" matches any set of characters, including the empty string.

    "**?**" matches any character.
  - The default contents of this field when the window opens depends on what element has focus when you choose **Search>Search**. In the example above, a model file had focus in the IDE, so ***.mod** is displayed as a default.
- **Scope** — you can expand or limit the scope of your search to search:
  - within a single file
  - within multiple files in the workspace
  - within the entire workspace
  - within previously-defined working sets
  - within previously-selected resources
  - within projects enclosing previously-selected resources

The Editing Area

This area displays various contents, depending on what you select.

- The contents of a model file in a text editor when you double-click a `.mod` file in the OPL Projects Navigator.
- The contents of a data file if you double-click a `.dat` file in the OPL Projects Navigator.
- The settings editor if you double-click an `.ops` settings file (see *Preferences and options* for details).
- Specialized ODM editors if you have IBM ILOG ODM installed and double-click on one of the files in the **ODM Application** area of a project.

Use the Editing Area to create new files, edit existing files, or examine active documents.

Note: Multibyte characters do not display in the IDE editor if your machine is not running on an operating system that supports such characters. See Internationalization .

Editing commands and shortcuts

The common commands you need to use while editing can be found on two menus:

- the **Edit** menu in the main menu bar, and
- the right-click context menu within the editor itself.

Keyword coloring and help

Within the files, language keywords are color-coded. The *Language Quick Reference* provides a quick reference to the keywords, which represent OPL and IBM ILOG Script instructions, functions, and methods.

Keywords are also connected to contextual help. To access it, select a keyword in the Editor and press **F1** to open the Help window and display help for that keyword. Once the window is open, clicking on any other keyword will display help for that keyword.

Additional information on working with the context-sensitive help can be found in the About the online help and Using the online help sections of *How to use the documentation*.

Customizing the editors

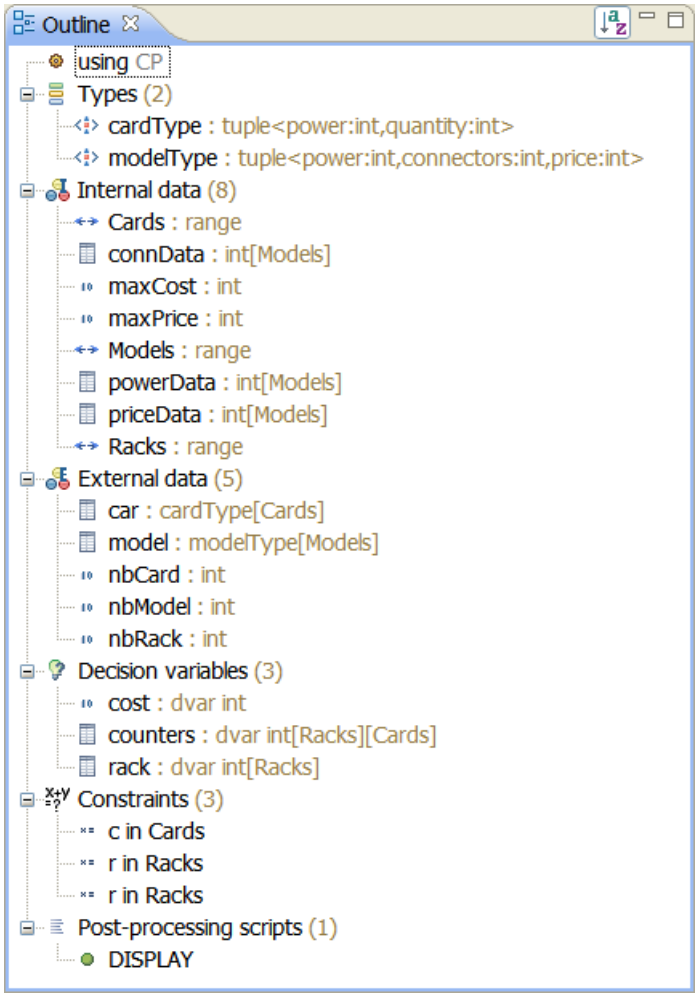
You can customize certain aspects of the Editing Area. Choose **Window>Preferences** and click either **Editor** in the **General** category, or **Colors** in the **OPL** category. See *IDE Preferences*.

Using the editor

For more information on the OPL editor, see *Editor features*.

The Outline

The **Outline** window displays the contents of the selected file as a tree. In the model outline, the main categories are sorted in their order of declaration. The lowest levels (leaves) are ordered alphabetically. The Outline window below shows the contents of the `config` model from the distributed CP example, with the preprocessing and postprocessing phases. Clicking an item highlights the corresponding line in the model.



The content of the Outline changes depending on the type of the selected file — model or settings.

- To display the outline of a model file, double-click the `.mod` file in OPL Projects Navigator to open it in the Editing Area. The first line of the Outline indicates whether the model

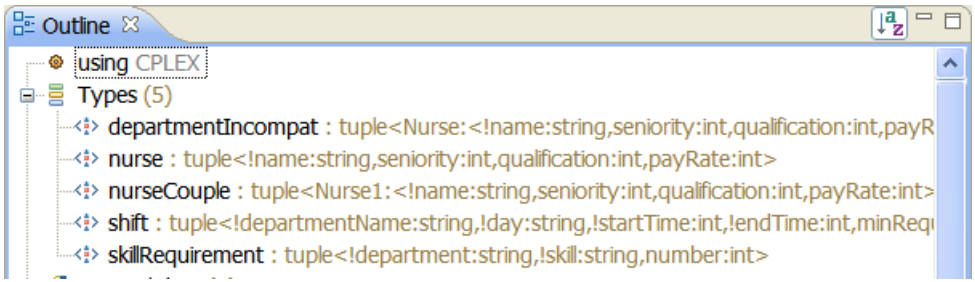
is to be solved by using CPLEX® or by using CP. The number of each model element (decision variables, constraints, scripting blocks, ...) is indicated in parentheses.

Tuple syntax in the Outline and error messages

The syntax used to present tuples and their components in the Outline window is as follows:

```
tuple<[!]name:type,...>
```

where **!** represents key components. This syntax is also used in OPL error messages when components of tuples are being referred to. For example, the following screenshot shows the tuples in the `Nurses` example:



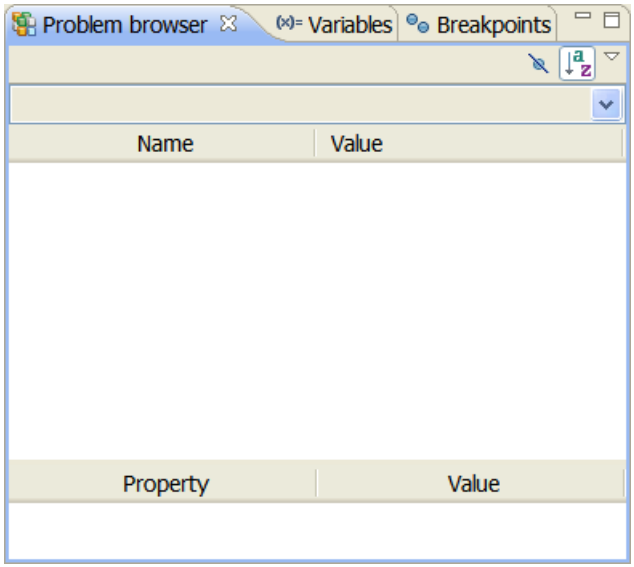
The Problem browser

The Problem browser window displays the model, data and solution values of the last configuration solved.

The information is presented in two panes with two columns each. The contents of the columns change when a run configuration is executed.

When opening a project

When a project has just been opened, the Problem browser is empty.

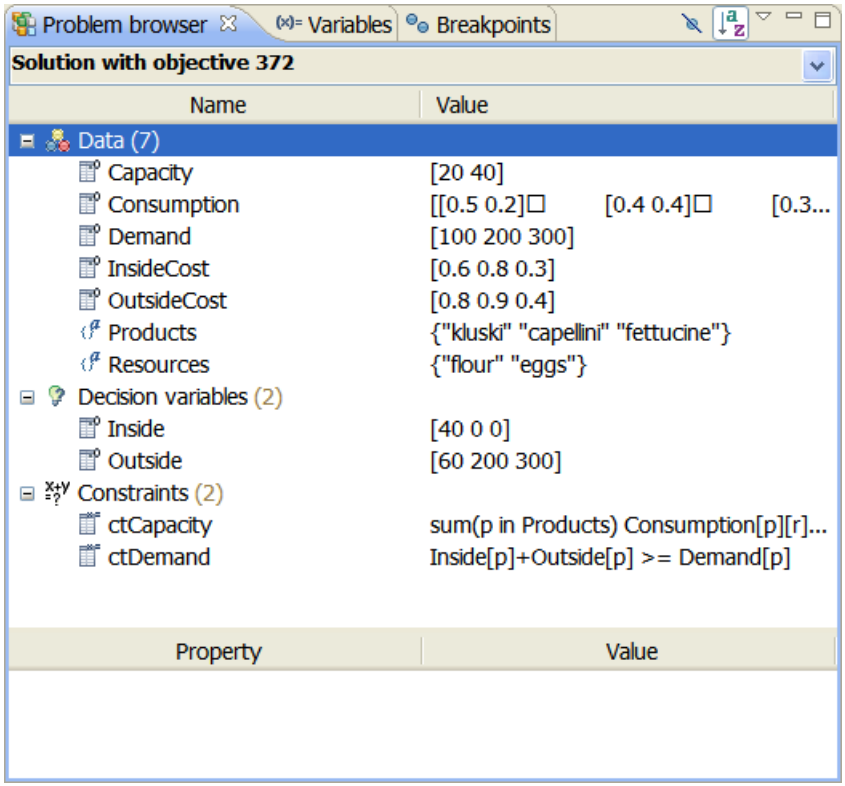


Empty Problem browser

- At the top, the drop-down list for solutions is empty and unavailable.
- In both the top and lower (Properties) pane, both columns are empty.

After running a project

After a run configuration is executed in Run mode, the Problem browser fills with data.



Problem browser after a solve

- At the top, the drop-down list for solutions displays a result.

  When solving problems with solution pools, multiple solutions are displayed in this drop-down list. Choosing one of them displays data for that solution in the lower part of the Problem browser.

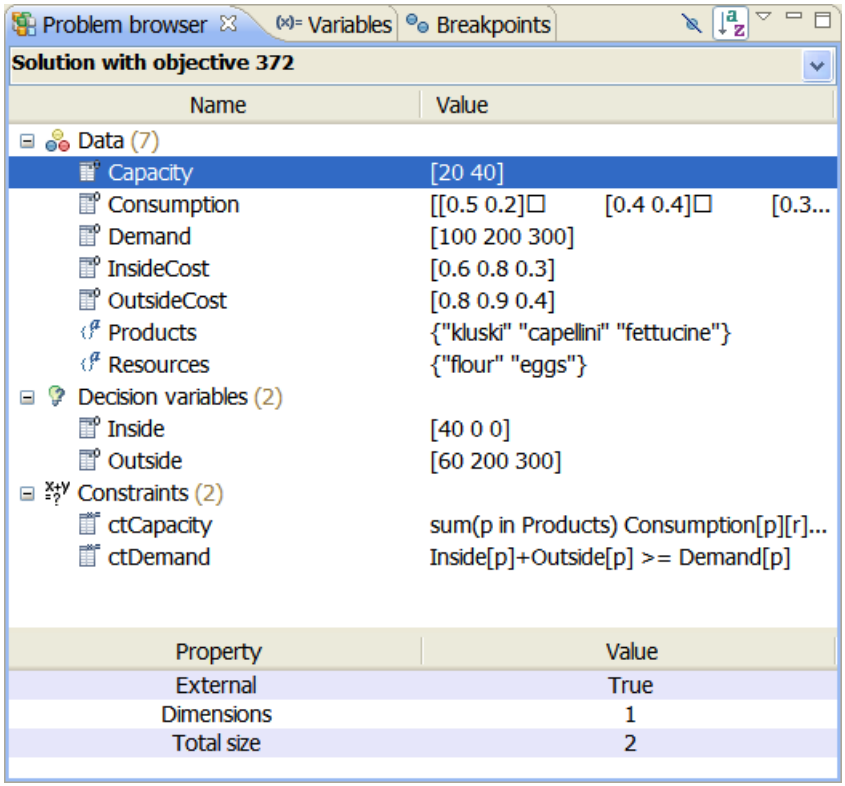
> **Note**: Feasible solutions can also be displayed in the Solutions tab if the options **Postprocess feasible solutions** and **Display solution** are selected in a settings file; see Setting language options.

- In the top pane, the **Name** column typically shows five categories of model elements: **Data** (types and constants), **Decision variables**, **Decision expressions**, **Constraints**, and **Post-processing** data, sorted in their order of declaration. The lowest levels (leaves) are ordered alphabetically.

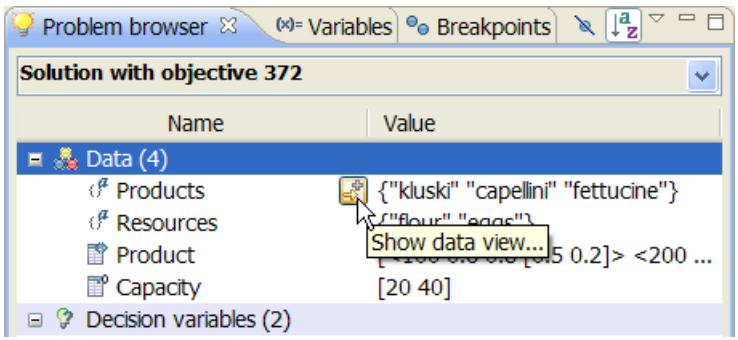
  The icons next to the objects show their types (see *Types in the Problem browser* in *IDE Reference*).

  The **Value** column now shows results for all elements.

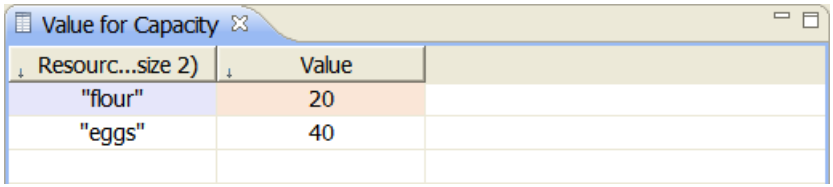
- If you select an item in the main Problem browser window (in this case, **Capacity**) properties for that object are shown in the lower part of the window:



- If you mouseover a data object (in this case, **Products**), a **Show data view** button appears (shown below with its tooltip visible):



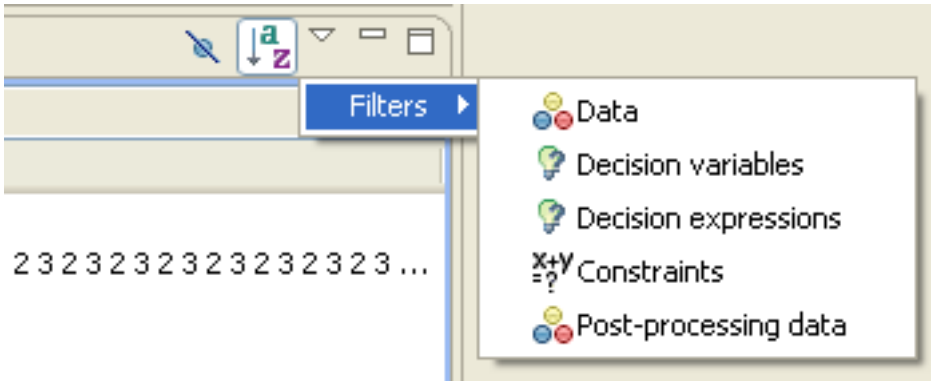
Clicking this button displays the data view in the main editing area:



Additional features of the Problem browser

Some of the features of the Problem browser are described briefly below:

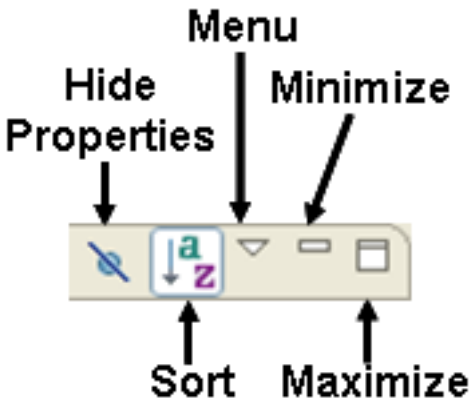
- **Hide Properties** button to conserve space.
- **Sort** button to sort .
- **View>Filters** menu allows filtering of displayed element types:



- Tooltips show data that is too wide to display.

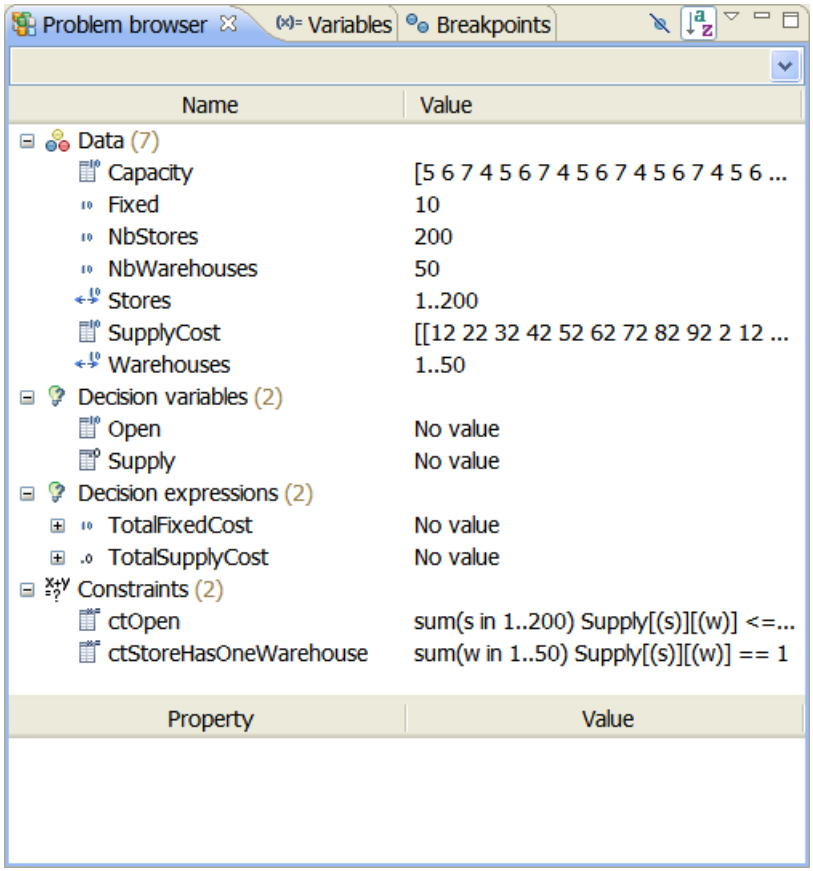
Problem browser toolbar

The Problem browser has a local toolbar. You can use its buttons to apply actions, or to customize, minimize, or maximize the view. Mouseover each icon to see a tooltip that describes its function.



Browsing before execution

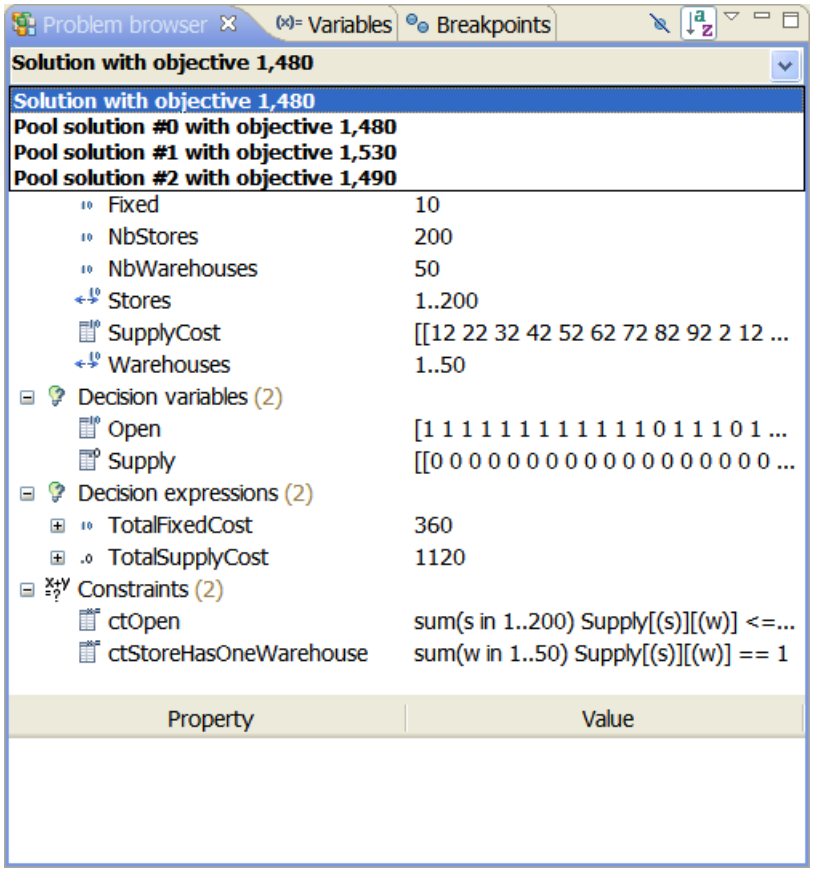
The **Browse** button  in the execution toolbar enables you to browse the model of the selected run configuration before execution. Notice that the solution list is empty and no value is available for decision variables, since no search for a solution has been performed.



Browsing before execution (`scalableWarehouse.mod`)

After execution

After a run configuration has been executed, the Problem browser displays values and a list of solutions as shown in the following screenshot.



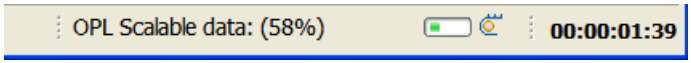
Problem browser after execution (`scalableWarehouse.mod`)

From the Problem browser, you can open views of data, variables, and constraints after execution.

See *Understanding the Problem Browser* in *Getting Started with the IDE* for more information.

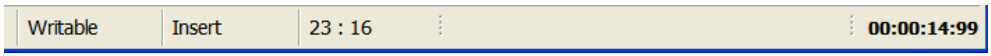
The Status Bar

As you solve, this area displays the execution status of projects being solved and the elapsed time of the solve.



Status Bar (partial view during a solve)

The Status Bar also shows the status of documents being edited in the Editing Area, and the current line and column number of the cursor.



Status Bar (partial view while editing)

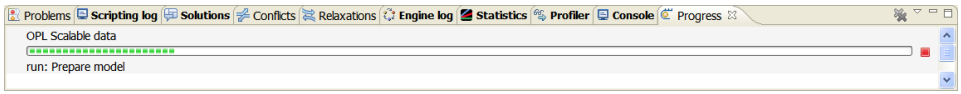
The middle part of the Status Bar indicates that the file currently being edited is **Writable** (as opposed to **Read-Only**) and that the editor is in **Insert** mode (as opposed to **Overwrite** mode). The numbers indicate the line number and column number of the current cursor location in the file.

The box at the right of the Status Bar displays an animated graphic while a project is running, and a message is displayed beside the graphic indicating progress.

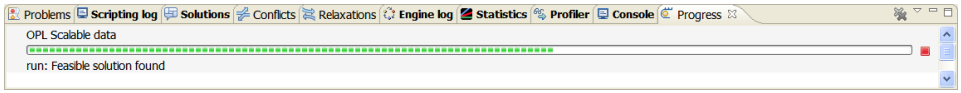
Run progress messages in the Status Bar

Whether the model is running in standard run mode or in debug, browse, or background mode, you can see messages about the progress of the run in the OPL IDE Status Bar:

- When the solve begins, a **Launching <run configuration name>** message appears at the right of the Status Bar.
- As the solve progresses, the message changes to **<run configuration name> <percent>**, with a percentage displayed to indicate progress.
- When the solve is finished, the message changes to **<run configuration name> 100%**, to indicate that the run has completed
- In addition, as the model is solving, the run indicator  becomes animated.
- If you click the **Shows background operations** icon  at the extreme right of the toolbar, a **Progress** tab appears in the Output Area. If the solve is still running, you see a display similar to the following:



When the solve has finished, you see a display similar to this:


Problems
Scripting log
Solutions
Conflicts
Relaxations
Engine log
Statistics
Profiler
Console
Progress
OPL Scalable data
run: Feasible solution found

The Output Area

This area is used by the IDE to return messages, error information, solutions, and results. This section provides a brief description of each tab or view. You will find more details in the hands-on tutorial *Getting Started with the IDE;* see *The Output tabs*.

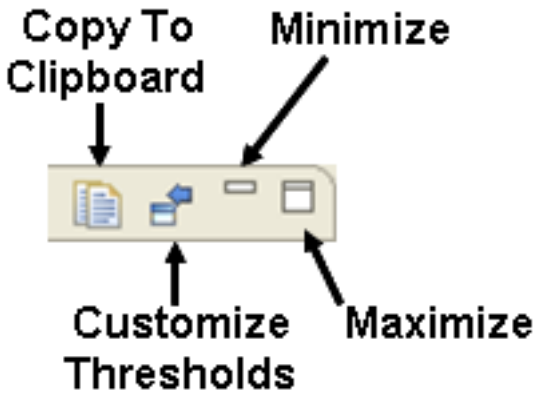
Brief description of Output Area tabs

Tab	Description
Problems	Syntax and semantic errors, and some execution errors (but not infeasibility). See *Problems* of *Getting Started with the IDE*.
Scripting log	Execution output related to scripting statements if the model or data file contains any. See *Scripting log* in *Getting Started with the IDE*.
Solutions	All feasible and final solutions to the optimization problem expressed by the run configuration. **Note:** Feasible solutions are displayed in the Solutions tab only if the options **Postprocess feasible solutions** and **Display solution** are selected in a settings file; see Setting language options. See also *Solutions* of *Getting Started with the IDE*.
Conflicts	For CPLEX® (MP) models only, conflicts found between constraints when the model is proved infeasible. See *Getting Started with the IDE*.
Relaxations	For CPLEX (MP) models only, relaxations of constraints or variables proposed to make the model feasible. See *Getting Started with the IDE*.
Engine Log	An account from the CPLEX or CP Optimizer solving engine. See *Engine Log* of *Getting Started with the IDE*.
Statistics	Figures about the algorithm used for execution and a progress chart of execution. See *Statistics* of *Getting Started with the IDE*.
Profiler	Time and memory used for the execution. See *Profiler* of *Getting Started with the IDE*.
Console	This tab appears when you launch an OPL run configuration in background mode using the External Tools button in the execution toolbar. See *Launching OPL run configurations in the background* for more information.

For additional information on the Output Area tabs, see the *The Output tabs* section in *Getting Started with the IDE*.

Output Area toolbar

The Output Area has a local toolbar. You can use its buttons to apply actions to the active view, or to customize, minimize or maximize the view. Mouseover each icon to see a tooltip that describes its function. (Not all of the icons below are present on every tab.)


Copy To Clipboard
Minimize
Customize Thresholds
Maximize

Toolbars

Describes the standard toolbar, the execution toolbar, and toolbars on individual views.

In this section

General description
Provides an overview of the different toolbars available in the main window of the IDE.

The standard toolbar
Provides information about the options available on the standard toolbar in the IDE.

The execution toolbar
Explains the options available on the execution toolbar in the IDE.

The Debug views toolbars
Explains the options available on the toolbars in the three debugging views in the OPL IDE — Debug view, Variables view, and Breakpoints view.

The OPL Projects Navigator view toolbar
Explains the toolbar options available on the OPL Projects Navigator view in the IDE.

The Problem browser view toolbar
Provides information about the options available on the Problem browser toolbar in the IDE.

The Output Area tab views toolbars
Provides information about the options available on the different tabs of the Output Area.

General description

Several toolbars are available in the main window, depending on what project element or view is active. Also, many views within the IDE have their own toolbars.

Many toolbar buttons are shortcuts to menu commands, but some buttons have no equivalent commands from the menu bar. When you move the pointer over a button in any toolbar, a tooltip displays a short description of it.

Also notice that some buttons are associated with a small arrow pointing downwards, such as the **Run** button  , **Debug** button  , and **Browse** buttons  . Click that arrow to select the element to which the button action must apply or to access other functions.

The main toolbars that are always visible at the top of the OPL IDE include:

- The standard toolbar
- The execution toolbar

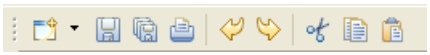
The toolbars on individual views include:

- The Debug view toolbar
- The OPL Projects Navigator view toolbar
- The Problem browser view toolbar
- Output Area tab views toolbars

Toolbars on other windows such as the Outline view or the Editing Area or the frame for the Output Area that contain only **Minimize** and **Maximize**  buttons are not described here, because their function is obvious.

The standard toolbar

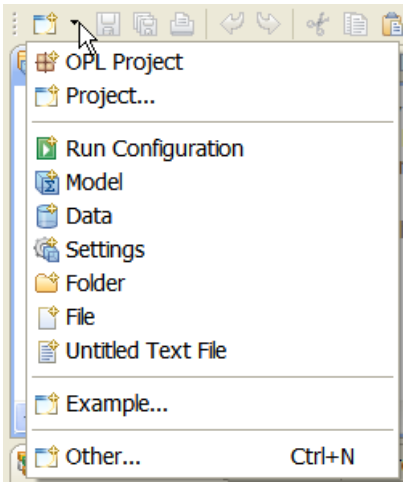
The standard toolbar presents a common set of buttons that are always available, although they may not all be active at the same time, as they are in the following screenshot.



Standard toolbar

Mouseover these buttons to see a tooltip for them. From left to right:

- **New** — clicking this button displays a wizard that allows you to select the type of new resource you want to create. Clicking the down arrow of the button reveals a drop-down menu of commands that you can use to create new elements in your OPL projects:



- **Save** – saves the currently-selected view (only active if the contents of the view have been changed).
- **Save All** – saves the all views whose contents have been changed (only active if the contents of one or more views have been changed).
- **Print** – sends the contents of the currently-active view to a printer dialog (only active if the contents of the view can be printed).
- **Undo** – undoes the previous action by reverting the change back to its previous state (only active if something has been changed).
- **Redo** – undoes the previous **Undo** action by reverting the change back to its previous state (only active if some change has been reverted using the **Undo** button).
- **Cut** – cuts the currently-selected text in editing view or other element to the Windows Clipboard for copying or pasting (only active when something has been selected).

- **Copy** – copies the currently-selected text in editing view or other element to the Windows Clipboard for pasting (only active when something has been selected).
- **Paste** – pastes the current contents of the Windows Clipboard to the current location (only active when something has been cut or copied).

See also *Menu commands and equivalent toolbar buttons*.

The execution toolbar

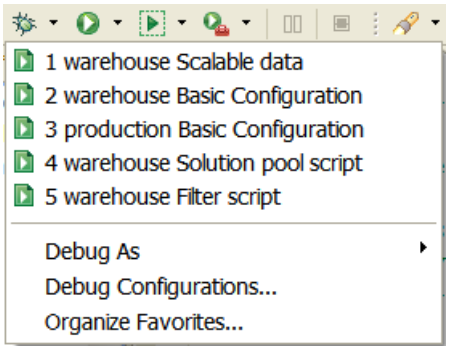
The execution toolbar presents a common set of buttons related to executing OPL projects. They are always available, although they may not all be active at the same time.

Execution toolbar

Mouseover these buttons in the IDE to see a tooltip for them. From left to right:

- **Debug** — clicking this button executes the last-executed run configuration and makes the Debug view and its toolbar functions .

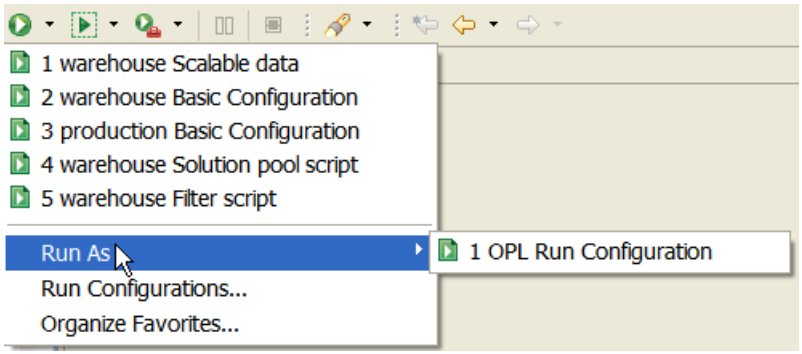
  Clicking the arrow of this button reveals a drop-down menu of commands that you can use to re-run recently-launched projects (the list at the top), select from a list of the run configurations of the current project (under the **Debug As** item), or open dialog windows to set up or organize preferred run configurations for debugging:



  For information on run status and progress messages, see *Run progress messages in the Status Bar*.

- **Run** — clicking this button executes the last-executed run configuration, as explained in*The Run options* section of *Introduction to the OPL IDE*.

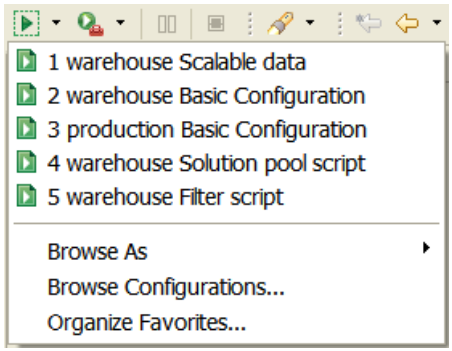
  Clicking the arrow of this button reveals a drop-down menu of commands that you can use to re-run recently-launched projects (the list at the top), select from a list of the run configurations of the project (under the **Run As** item as shown below), or open dialog windows to set up or organize preferred run configurations for running:



For information on run status and progress messages, see *Run progress messages in the Status Bar*.

- **Browse** — clicking this button executes the last-executed run configuration in Browse mode. That is, the project is loaded into memory so that its elements can be browsed using the Problem browser, but without solving it.

  Clicking the arrow of this button reveals a drop-down menu of commands that you can use to re-run recently-launched run configurations (the list at the top), select from a list of the run configurations (under the **Browse As** item as shown below), or open dialog windows to set up or organize preferred run configurations for browsing:

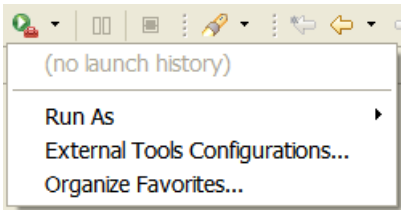


  For information on run status and progress messages, see *Run progress messages in the Status Bar*.

- **External Tools** — clicking this button executes the last-executed launch configuration in background mode. That is, when using this button to run an OPL model in the background, the run configuration is launched using `oplrun` and solved in the background, and displays the output in its own **Console** tab in the Output Area as it runs.

  You can continue to work editing the same project or a different project while the External Tools run is continuing in the background.

  Clicking the arrow of this button reveals a drop-down menu of commands that you can use to re-browse recently-launched projects (the list at the top), select from a list of the run configurations (under the **Run As** item), or open dialog windows to set up or organize preferred launch configurations for running as background jobs:



For instructions on how to set up launch configurations using the External Tools button, the options available to you, and the Console view in the Output Area where you view output and control the processing of the background process, see *Launching OPL run configurations in the background*.

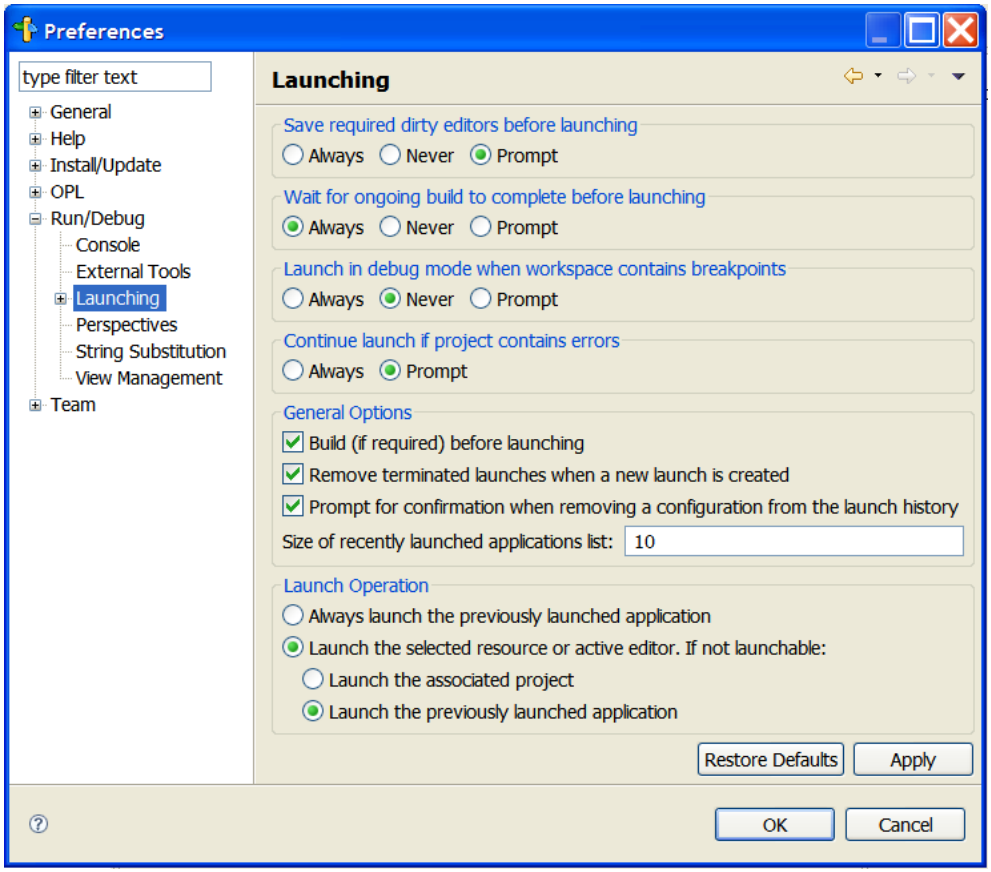
- **Pause/Resume** – a toggle button that pauses/continues the current run (only active if a run is in progress).
- **Abort** – aborts the current run (only active if a run is in progress).

For more information, see:

- *What happens when you execute a run configuration*.
- *Understanding solving statistics and progress (MP models)* and *Understanding solving statistics and progress (CP models)* in *IDE Tutorials*.
- *Using IBM ILOG Script for OPL* in *IDE Tutorials*.

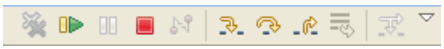
Run, Debug, Browse, and External Tools button options

The default behavior of the **Run**, **Debug**, **Browse**, and **External Tools** buttons can be configured using the **Run/Debug > Launching** section of the **Window > Preferences** editor:


Preferences
type filter text
General
Help
Install/Update
OPL
Run/Debug
Console
External Tools
Launching
Perspectives
String Substitution
View Management
Team
Launching
Save required dirty editors before launching
Always Never Prompt
Wait for ongoing build to complete before launching
Always Never Prompt
Launch in debug mode when workspace contains breakpoints
Always Never Prompt
Continue launch if project contains errors
Always Prompt
General Options
Build (if required) before launching
Remove terminated launches when a new launch is created
Prompt for confirmation when removing a configuration from the launch history
Size of recently launched applications list: 10
Launch Operation
Always launch the previously launched application
Launch the selected resource or active editor. If not launchable:
Launch the associated project
Launch the previously launched application
Restore Defaults
Apply
OK
Cancel

The Debug views toolbars

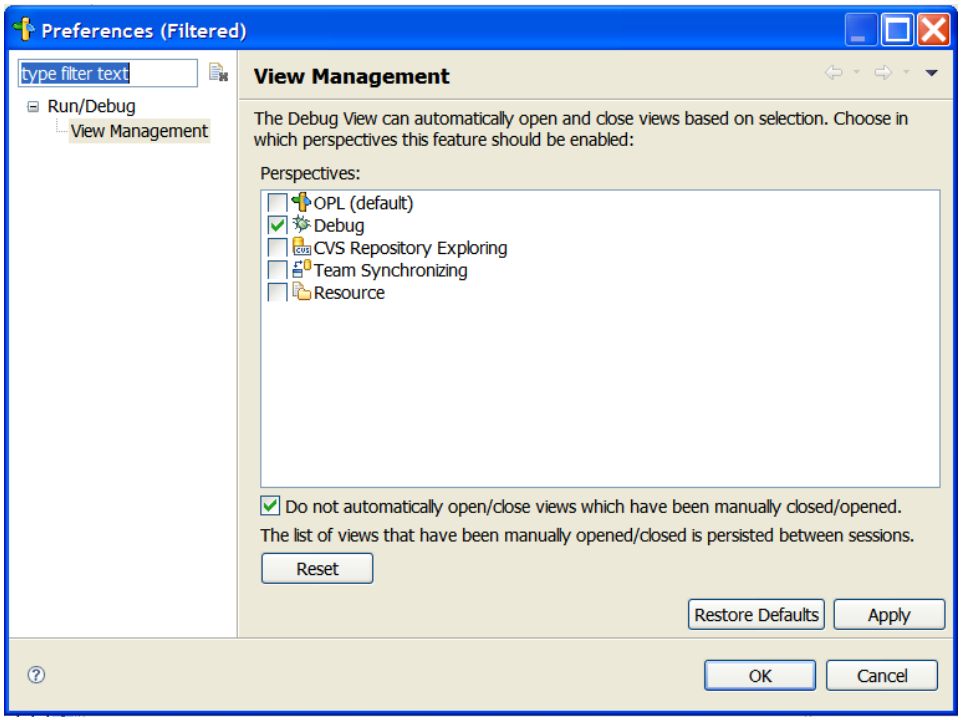
The three debugging views — Debug view, Variables view, and Breakpoints view — become active when the IDE starts executing a project. They remain active as long as execution is in progress, including when a run is suspended. You can use the buttons in the toolbars of each view to control execution.



Debug view toolbar

Mouseover these buttons to see a tooltip for them. From left to right:

- **Remove All Terminated Launches** – clears the window of runs that have completed or have been terminated (only active if the previous runs are visible in the view).
- **Pause/Resume** – a toggle button that pauses/continues the current run (only active if a run is in progress).
- **Suspend** – suspends the current run (only active if a run is in progress).
- **Terminate** – terminate the current run (only active if a run is in progress, and remains active if the run has been stopped using the **Abort** button on the execution toolbar).
- **Disconnect** – terminates the connection between the debugger and the remote debug target (only active if a run is in progress).
- **Step Into** – when debugging, resumes the run and steps into the next method call at the currently executing line of code (only active during debugging, with breakpoints set).
- **Step Over** – when debugging, resumes the run and steps over the next method call at the currently executing line of code without executing it (only active during debugging, with breakpoints set).
- **Step Return** – when debugging, resumes the run returns from a method that has been stepped into (only active during debugging, with breakpoints set).
- **Drop to frame** – when debugging, re-enters the selected stack frame in the Debug view (not active in OPL).
- **Use Step Filters** – when debugging, activates/deactivates preselected step filters that have been set up for use in the Debug view (not active in OPL).
- **Menu** – opens a View Management dialog box for setting preferences in Debug view.

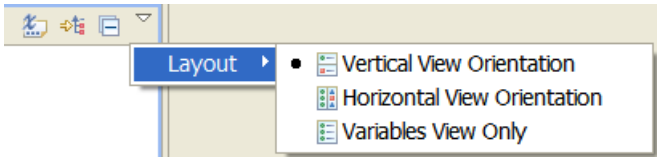


Note: You set breakpoints for debugging by double-clicking in the left margin of the editor. Click a breakpoint to temporarily disable it; double-click to remove it.

Variables view toolbar

Mouseover these buttons to see a tooltip for them. From left to right:

- **Show Type Names** – displays/hides type names for the elements in Variables view (not active when columns are displayed).
- **Show Logical Structure** – displays/hides type logical structures in Variables view.
- **Collapse All** – collapses all expanded elements in the view (only active if the elements have been expanded).
- **Menu** – opens a Layout menu for setting display preferences in Variables view.

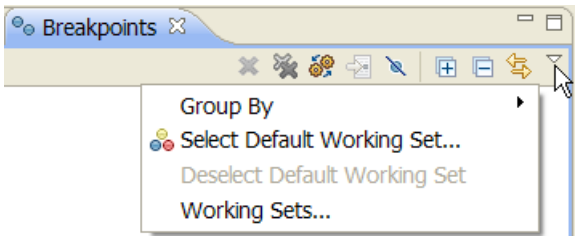


- **Minimize** – minimizes the view.
- **Maximize** – maximizes the view.

Breakpoints view toolbar

Mouseover these buttons to see a tooltip for them. From left to right:

- **Remove Selected Breakpoints** – delete the currently-selected breakpoints from the view.
- **Remove All Breakpoints** – delete all current breakpoints from the view.
- **Show Breakpoints Supported By Selected Target** – displays/hides breakpoints supported/not supported by the currently-selected debug target.
- **Go To File For Breakpoint** – open the associated resource for the breakpoint, make it active, and highlight the location of the breakpoint. (not active in OPL).
- **Skip All Breakpoints** – mark all breakpoints in the current view as skipped.
- **Expand All** – expands all collapsed elements in the view (only active if the elements have been collapsed).
- **Collapse All** – collapses all expanded elements in the view (only active if the elements have been expanded).
- **Link With Debug View** – updates Breakpoints view with information from Debug view.
- **Menu** – opens a menu for setting preferences in Breakpoints view.



Note: Three other debugging views are shown in the Show View popup window — Expressions view, Memory view, and Registers view. These are not active in OPL.

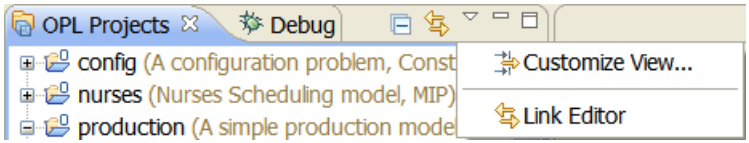
The OPL Projects Navigator view toolbar

The OPL Projects Navigator view toolbar presents the following sets of buttons.

OPL Projects Navigator toolbar

Mouseover these buttons to see a tooltip for them. From left to right:

- **Collapse All** – collapses all expanded elements in the view (only active if the elements have been expanded).
- **Link open editors with content in the Navigator** – when you have multiple files open for editing, you can configure the OPL Projects Navigator view to automatically bring an open file to the foreground (make its editor session the active editor).
- **Menu** — clicking this button reveals a drop-down menu of commands that you can use to create this view:



- **Minimize** – minimizes the view.
- **Maximize** – maximizes the view.

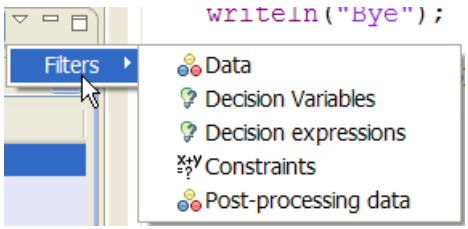
The Problem browser view toolbar

This toolbar allows you to perform functions on the problem being currently displayed in Problem browser view.

Problem browser toolbar

Mouseover these buttons to see a tooltip for them. From left to right:

- **Hide Properties** – hides the properties section of this view.
- **Sort** – sorts the contents of the view.
- **Menu** — clicking this button reveals a drop-down menu of commands that you can use to filter this view:



- **Minimize** – minimizes the view.
- **Maximize** – maximizes the view.

The Output Area tab views toolbars

The toolbar options available on the different tabs of the Output Area vary from tab to tab. Some exist only on one tab, others exist on several different tabs.

Mouseover these toolbar buttons to see tooltips for them.

- **Clear View/Log/Console**  – on the **Script log**, **Solutions**, and **Engine log** tabs, clears the current contents of the view.
- **Configure Filters**  – on the **Problems** tab, allows you to set filter preferences.
- **Copy To Clipboard**  – on the **Conflicts**, **Relaxations**, and **Profiler** tabs, copies the current contents of the view to the Windows clipboard.
- **Customize Thresholds**  – on the **Profiler** tab, allows you to set threshold preferences.
- **Detailed memory format**  – on the **Profiler** tab, toggles the display to show more details of memory usage.
- **Display Selected Console**  – on the **Console** tab, opens a view of the type selected from the drop-down list in the Output Area.
- **Maximize**  – maximizes the view.
- **Menu**  — on the **Problems** tab, reveals a drop-down menu specific to the tab.
- **Minimize** – minimizes the view.
- **Open Console**  – on the **Console** tab, opens a view of the type selected from the drop-down list in the Output Area.
- **Pin Console**  – on the **Console** tab, ensures that the current **Console** tab remains on top (for example, if you had multiple **Console** tabs open)
- **Remove All Terminated Launches**  – on the **Console** tab, removes all terminated launches from the view.
- **Remove Launch**  – on the **Console** tab, removes the current launch from the view.
- **Scroll Lock**  – on the **Script log** and **Engine log** tabs, stops the scrolling of output to the window.
- **Show Console When Standard Error Changes**  – on the **Console** tab, displays the tab when the process writes to standard error.
- **Show Console When Standard Out Changes**  – on the **Console** tab, displays the tab when the process writes to standard output.

- **Terminate**  – on the **Console** tab, this button becomes red, and can be used to abort the run.

The text editor

Describes the editing features available in the Editing Area, explains how to switch between Editor windows and how to resize an Editor window, and provides keyboard shortcuts.

In this section

Editor features
Provides a table of the features available in the OPL IDE editor.

Switching between Editor windows
Explains how to edit several documents at once in the IDE, and switch between them.

Resizing an Editor window
Explains how to resize the editor window.

Keyboard shortcuts
Lists the keyboard shortcuts available to you in the OPL IDE.

Model Completion and Model Templates
Describes the features of the OPL IDE editor that provide assistance when completing your OPL models.

Local History and its related features
How to track and compare different versions of your files as you edit them in the OPL IDE.

'Compare With' features
How to compare files with each other and with Local History.

'Replace With' features

How to compare files with each other and replace the contents from other versions of the file in Local History.

Editor features

The *IDE editor features* table lists the IDE Editor features.

IDE editor features

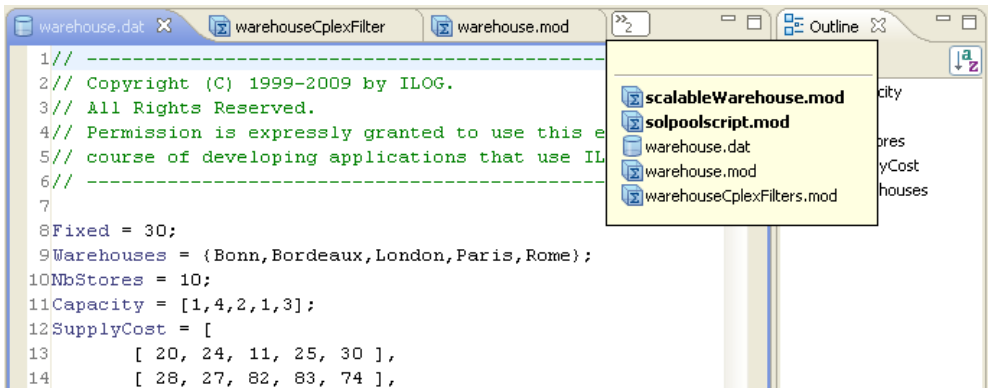
Editor feature name	Description
Syntax coloring	The syntax in model and data files is colored differently, according to its type. The color scheme is customizable, see *IDE Preferences*.
Multiple levels of Undo and Redo	You can undo and redo your modifications without any limit.
Automatic indentation	Once an indentation block has been started, it continues on successive lines. You can increase or decrease the indentation depth by setting the Tabulation size (see *IDE Preferences*).
Bracket (or brace) matching	When typing **]**, **}** or **)**, the matching opening bracket is highlighted for 800 ms. In data files, **<** and **>** are also matched.
Margin symbols	The Editor has a left margin that can contain symbols, such as: ♦ the red box that indicates an error  ♦ the blue circle that indicates a breakpoint  in debugging mode ♦ the blue circle with an arrow that indicates a breakpoint  in debugging mode ♦ the white arrow that indicates the current line  in debugging mode. **Note**: You set breakpoints for debugging by double-clicking in the left margin of the editor. Click a breakpoint to temporarily disable it; double-click to remove it.
Reload prompt	If you modify a file with an external editor, you are prompted to reload the file as soon as the IDE Editor regains focus.
Customization	You can set options to customize the Editor, see *IDE Preferences*.

Switching between Editor windows

You open your model files, data files and settings files for editing by double-clicking on the file in OPL Projects Navigator.

Multiple editors can be open in the Editing Area at once. You can switch back and forth between the open views by clicking the tabs of the views that are visible in the Editing Area.

If so many edit views are open that all of their tabs cannot be displayed, a “more views” icon  becomes visible. Click it and a list of the other views appears:



Click any of the views in the popup list to view them.

Resizing an Editor window

An Editor window may be minimized or maximized. By default, it is maximized. The buttons on the right side of the menu bar  enable you to minimize the window or restore it to normal size.

To close an Editor window, click the **X** icon in the window's tab  .

By double-clicking on the tab of the Editor window, you can temporarily expand it to the size of the IDE frame, and thus use almost all the screen for editing. Double-click the tab again to return it to its normal size.

Keyboard shortcuts

Editor functions lists the commands available in the text editor of the IDE Editing Area.

Note:
- **Ctrl** = Control
- **Ctrl + X** = Control and X key pressed simultaneously

Editor functions

Action	Visual Mode
Show shortcuts	**Ctrl + Shift + L**
Next open edit window	**Ctrl + Shift + F6**
Previous view	**Ctrl + Shift + F7**
Save	**Ctrl + S**
Save All	**Ctrl + Shift + S**
Cut	**Ctrl + X** or **Shift + Delete** on selection
Copy	**Ctrl + C** or **Ctrl + Insert**

Action	Visual Mode
	on selection
Paste	**Ctrl + V**, or **Shift + Insert**
Undo	**Ctrl + Z**
Redo	**Ctrl + Y**
Uppercase selected text	**Ctrl + Shift + X**
Lowercase selected text	**Ctrl + Shift + Y**
Next character (right)	Right arrow key
Previous character (left)	Left arrow key
Next line (down)	Down arrow key
Previous line (up)	Up arrow key
Next screen (down)	Page Down key
Previous screen (up)	Page Up key
Beginning of file	**Ctrl + Home**
End of file	**Ctrl + End**
Next word (right)	**Ctrl** + right arrow key
Previous word (left)	**Ctrl** + left arrow key
Beginning of line	**Home**
End of line	**End**
Delete character after cursor	**Delete**
Delete character before cursor	**Backspace**
Scroll down one line	**Ctrl** + down arrow key
Scroll up one line	**Ctrl** + up arrow key
Recenter on current line	**Ctrl + L**

Action	Visual Mode
Mark the beginning of a selection	Click left mouse button and drag cursor, or use arrow keys
Mark the end of a selection or extend a selection	Release left mouse button, or press **Shift** + left arrow key, or **Shift** + arrow keys, or **Shift + Page** keys, or **Ctrl + Shift** + arrow keys, or drag mouse
Stop command in progress	Release selection
Select line	Triple-click the line.
Select word	Double-click the word.
Select all	**Ctrl + A** or quadruple-click
Switch edit windows	Keep **Ctrl** down and press **Tab** successively
Indent a region	Context menu command only
Unindent a region	Context menu command only
Indent at location	Context menu command only
Unindent at location	Context menu command only
Comment / Uncomment lines	**Ctrl + Y** (toggle)
Find	**Ctrl + F**
Find Next	**Ctrl + K**
Find Previous	**Ctrl + Shift + K**
Find Incremental Next	**Ctrl + J**
Find Incremental Previous	**Ctrl + Shift + J**
Word Completion	**Alt + /**
Join Lines	**Ctrl + Alt + J**
Go to a specific line	**Ctrl + L**

Model Completion and Model Templates

Describes the features of the OPL IDE editor that provide assistance when completing your OPL models.

In this section

Overview
Provides an overview of the Model Completion and Model Templates features in the OPL IDE editor.

Model Completion
Describes how to use the Model Completion feature in the OPL IDE editor.

Model Templates
Describes how to use the Model Templates feature in the OPL IDE editor.

Overview

Model Completion is a feature of the OPL IDE editor that allows it to prompt you during the creation of an OPL model, and allows you to select OPL and OPL Script keywords and functions from a context-sensitive list, and insert them into your model. You access this level of code assistance by pressing **Ctrl + space** in the OPL IDE editor.

Another level of code assistance is available when 1) your model contains tuples, and 2) an Outline is present for the model (that is, there are no syntax errors in the saved model). This level of code assistance prompts you with variables and tuple component names from the model Outline and is accessed using a dot syntax. That is, you access it by typing a single period or dot (**.**) in the OPL editor.

Model Templates is a related feature of the OPL IDE editor that allows you to select longer snippets of code, along with placeholders for required variables, from a context-sensitive list, and insert them into your model. As with Model Completion, you access this by pressing **Ctrl + space** in the OPL IDE editor.

The elements available (displayed in the popup window) for Model Completion are supplied with OPL. The elements available for Model Outline Completion are supplied by your OPL model.

With regard to templates, a number of predefined templates are supplied with OPL, but you can edit them or add your own, as you see fit.

All of these features are discussed in the following sections.

Model Completion

The Model Completion feature of the OPL IDE editor affords many code assistance benefits to you as you write your OPL models.

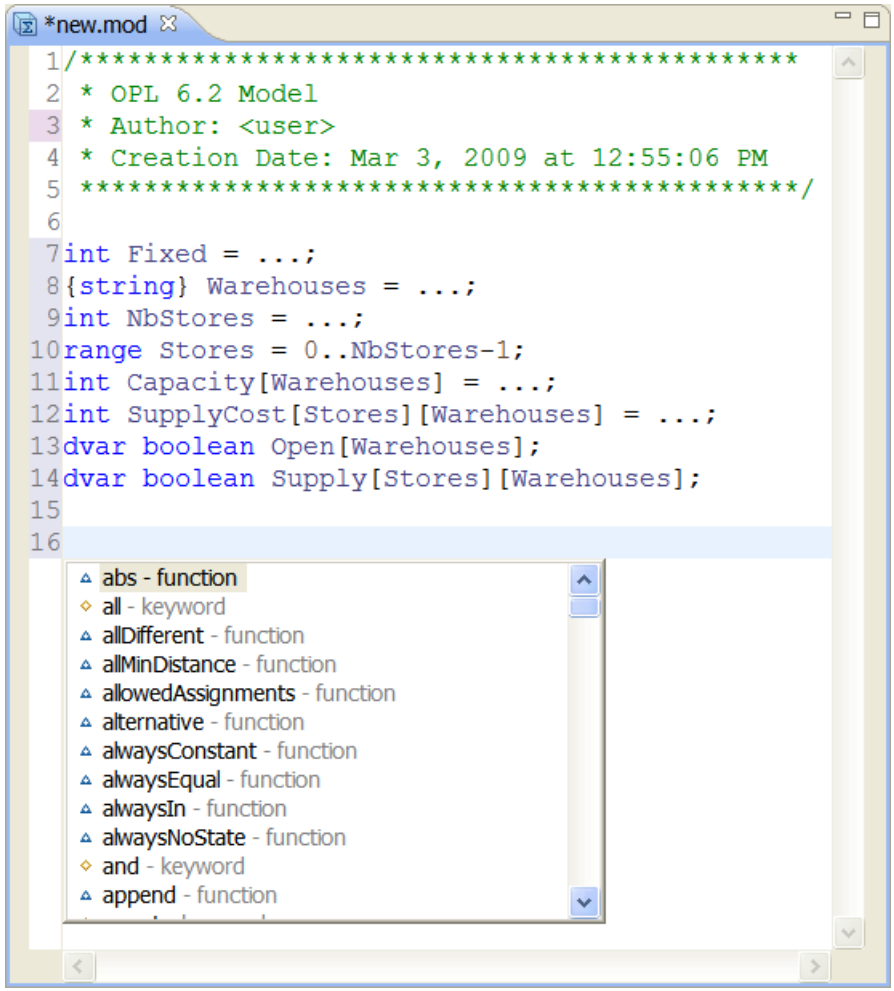
- With one keystroke (**Ctrl + space**), you can switch between normal code entry and a popup window that prompts you with OPL and OPL Script keywords and functions to insert into your model.
- Using the dot (**.**) syntax, you can also display a popup window that prompts you with variables and tuple component names from your model Outline.
- The list of keywords and functions is *context sensitive* in that it can detect whether you are writing a CPLEX® or CP Optmizer model, and prompt you only with the appropriate code elements. It can also detect whether you are entering OPL code or OPL Script code, and display an appropriate list of code elements.
- Before or after opening the Model Completion popup window by pressing **Ctrl + space**, typing the first character or characters of the keyword or function you are searching narrows the display in the popup window to only those that match what you type.
- Selecting keywords and functions from a predefined list reduces typing. It can also help you to avoid spelling or typing errors, and ensures that entries are made with the proper capitalization (for example, `allDifferent` as opposed to `AllDifferent` or `alldifferent`).
- If you are searching for a keyword or function that you haven't used for a while, you can search the list to jog your memory.

Working with Model Completion

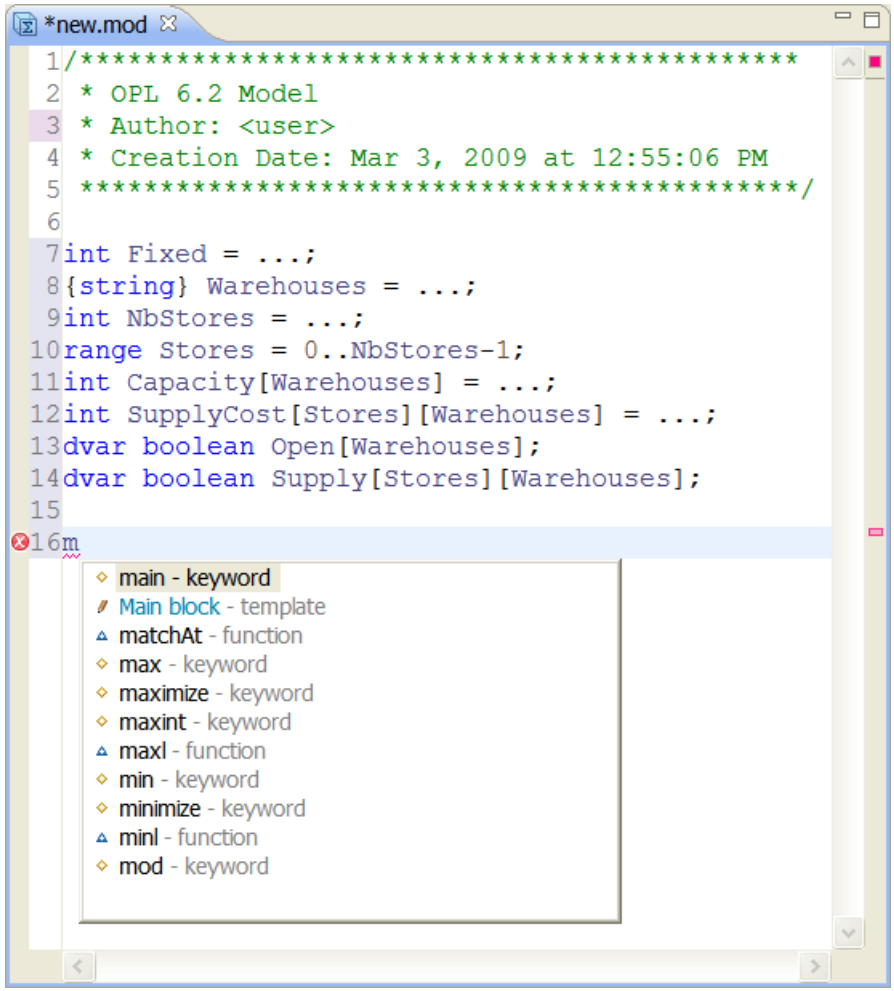
Model Completion can be invoked at any time in the OPL IDE editor while creating or editing an OPL model.

1. **To display the Model Completion popup window**, press **Ctrl + space**.

   The popup window will open, displaying entries that are appropriate for the context. For example:

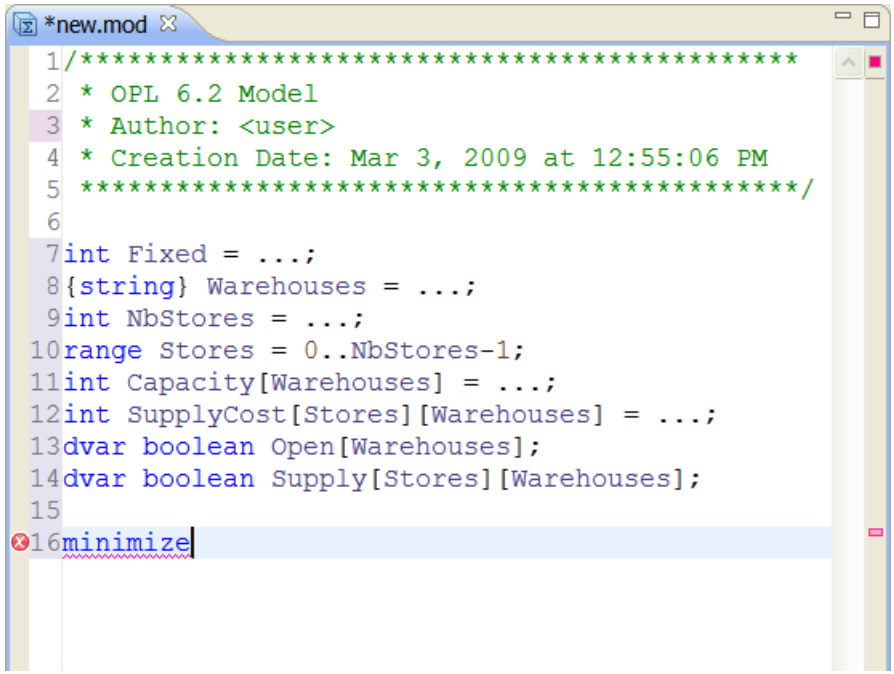


2. **To filter the display**, type one or more characters of the keyword or function you are searching for, before or after pressing **Ctrl + space**. For example, here the user types the letter **m**.



See the *Examples* section below for more suggestions on ways to use this feature.

3. **To insert the selection**, double-click it or select it and press **Enter**. In this example, the user has selected the `minimize` keyword, and it is inserted into the editor at the cursor position.

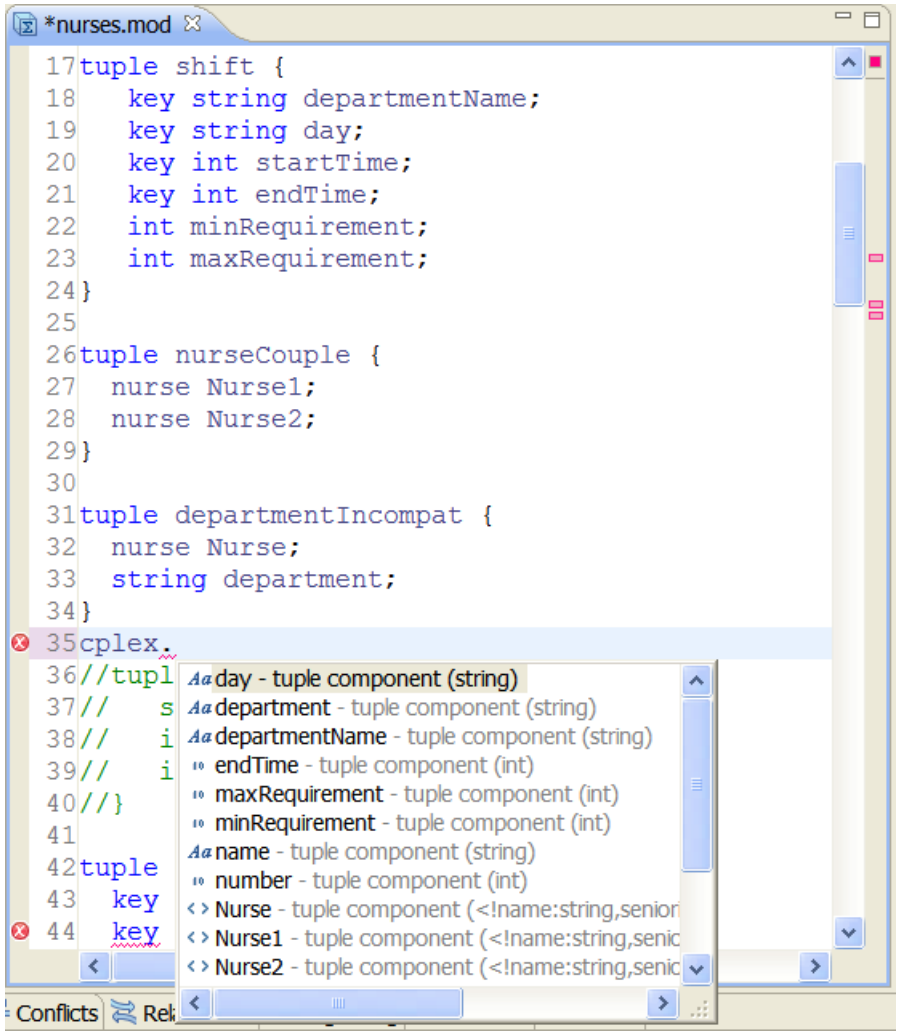


Note that the red **X** symbol appears in the left margin of the editor at this point, because `minimize` has required arguments that have not yet been supplied.

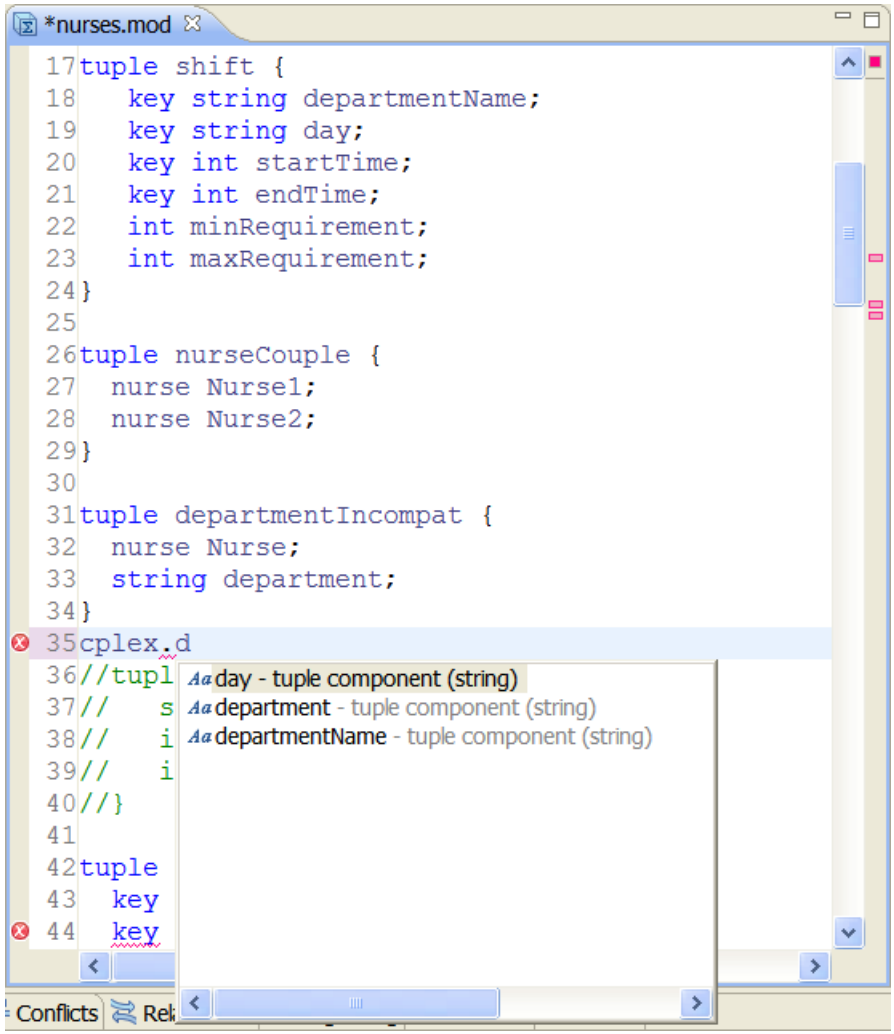
4. **To use the dot sequence method**, type a single period or dot (**.**) in the OPL editor.

Note: This level of code assistance is available to you only if 1) your model contains tuples, and 2) a validated Outline exists for your model.

For example, in the screenshot below the user has typed **cplex.** (the `cplex` keyword followed by a single period or dot) in the model file for the `nurses` example, which contains tuples. The resulting popup window displays a list of tuples and tuple components taken from the `nurses.mod` Outline, any of which can be inserted.



The user could continue to filter the list by typing additional characters after the **cplex.** entry. For example, typing **cplex.d** filters the list to show only those tuple components in the model that start with the letter **d**.

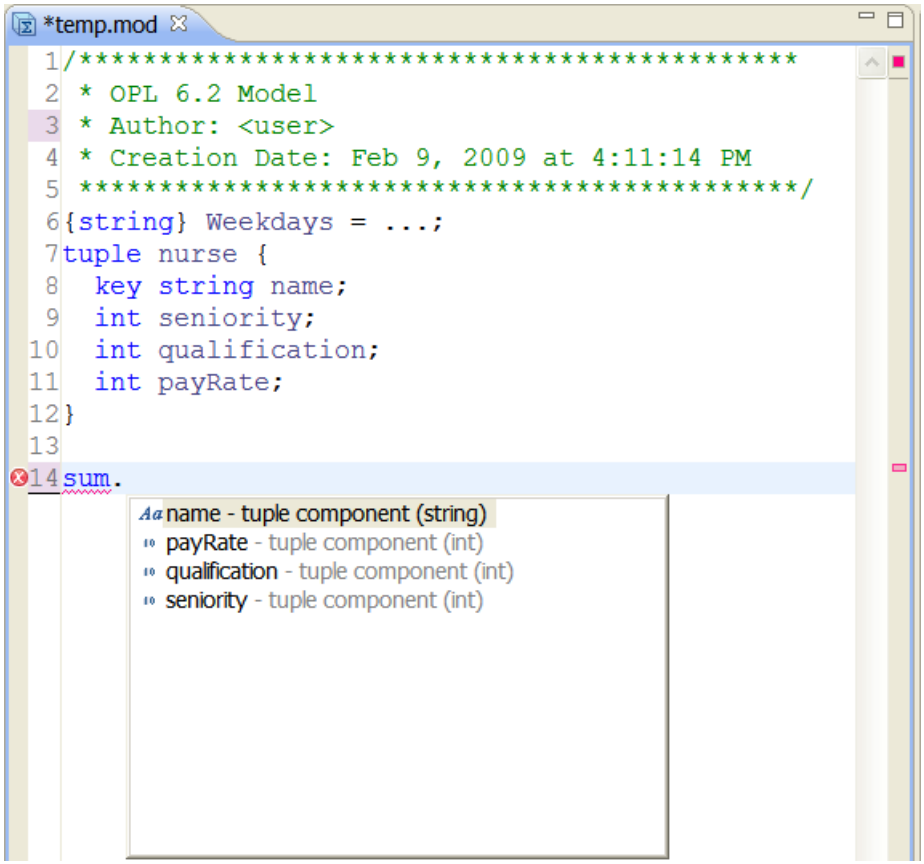


You can continue in this fashion as you write your model, pressing **Ctrl + space** or a single period or dot (**.**) at any point for code assistance while writing your model. See the *Examples* section below for more suggestions on ways to use these features.

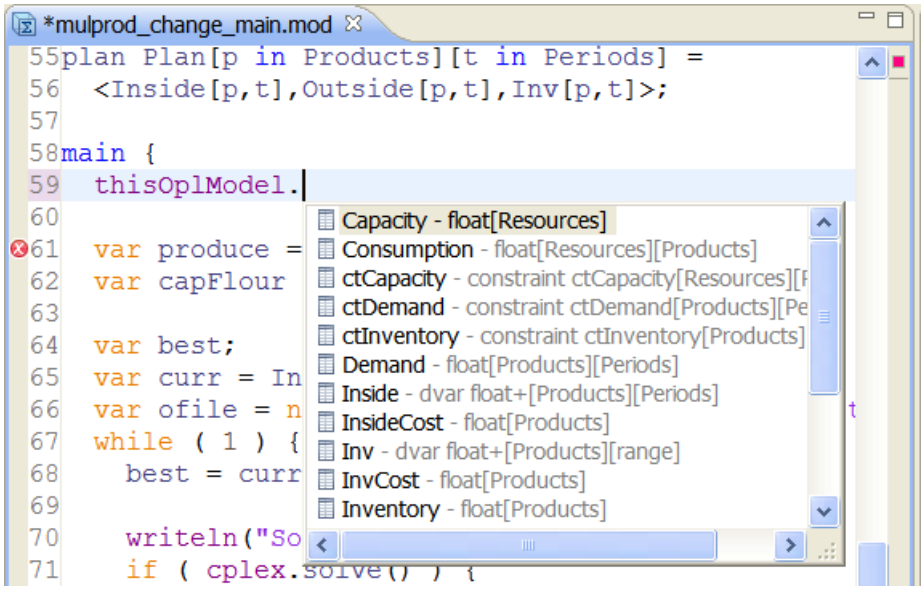
Examples

Some additional examples of ways you can use Model Completion are suggested below:

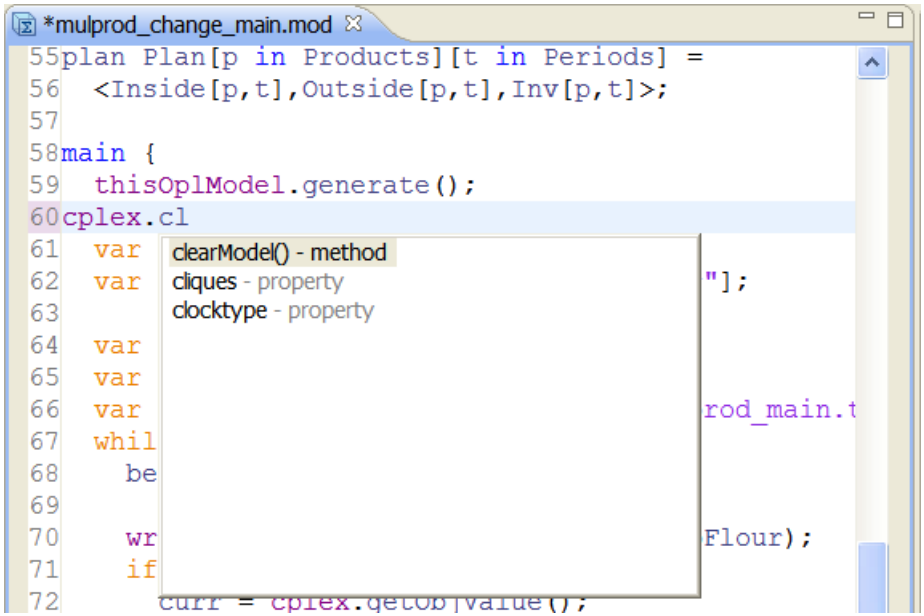
- After defining a tuple, you want to insert one of its components after a `sum` keyword. In this example, the user has typed **sum.**



♦ In this example, the user has typed **thisOplModel.**



♦ In this example, in a scripting context, the user has typed **cplex.cl**



If the user were to type an additional **o** to limit the list to `clocktype` and press **Enter**, what would be inserted into the editor would be **cplex.clocktype**.

Model Templates

The Model Templates feature of the OPL IDE editor affords additional benefits to Model Completion.

- Templates (snippets of code), either predefined and supplied with OPL or created by you, can be selected from a list and inserted in their entirety.
- They can provide an easy way to insert a complete structure, including placeholders for variables that can then be replaced with the variables by tabbing between them and typing or using the Dot (**.**) syntax to access variables and other elements from your model.
- Templates can be *context sensitive*, allowing different completions for different parts of the edited file. For example, entering OPL code vs. entering OPL Script code.
- Templates can be simple constructions, such as:

  ```
  using ${engine}
  ```

  They can be more complex, with line feeds, for example:

  ```
  execute ${name} {
  }
  ```

  They can include `dvar` arrays inside the model (the number of dimensions being automatically correct). For example:

  ```
  x[${i}][${j}]
  ```

- A number of useful templates are supplied with OPL, but you can add your own as you find uses for them.
- All templates can be edited using the OPL Preferences editor. New templates can be easily created, based on code that is frequently reused in your work.

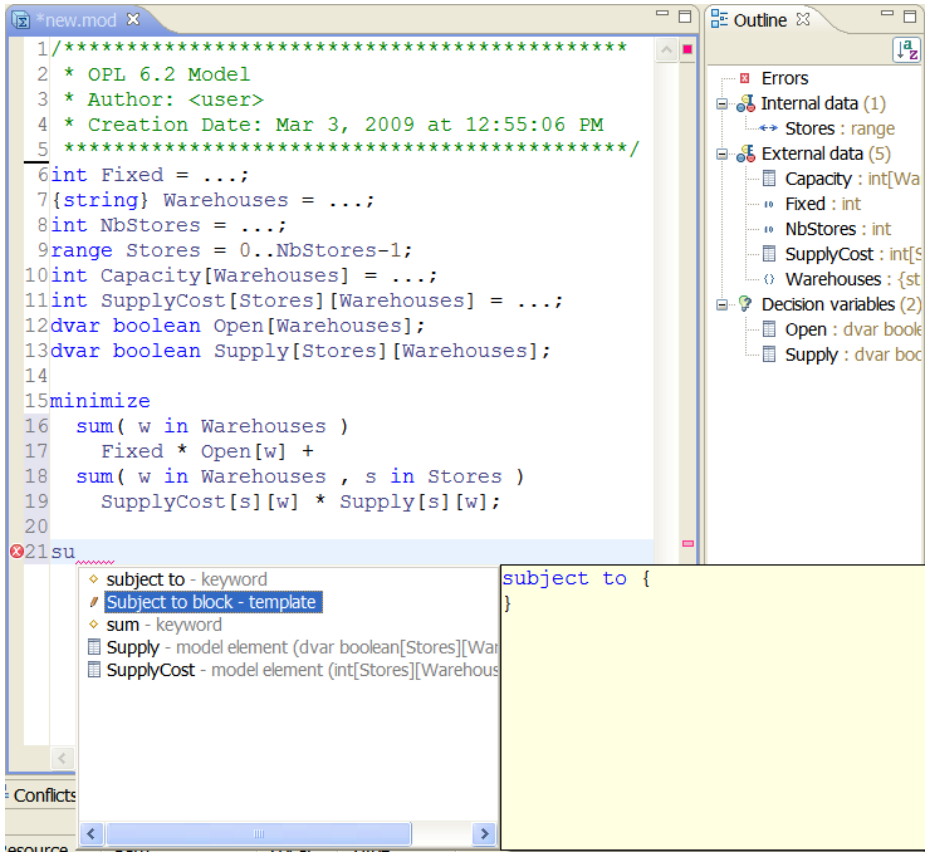
Working with Model Templates

Model Templates can be selected from the Model Completion popup window any time it is open.

1. **To display the Model Completion popup window**, press **Ctrl + space**.

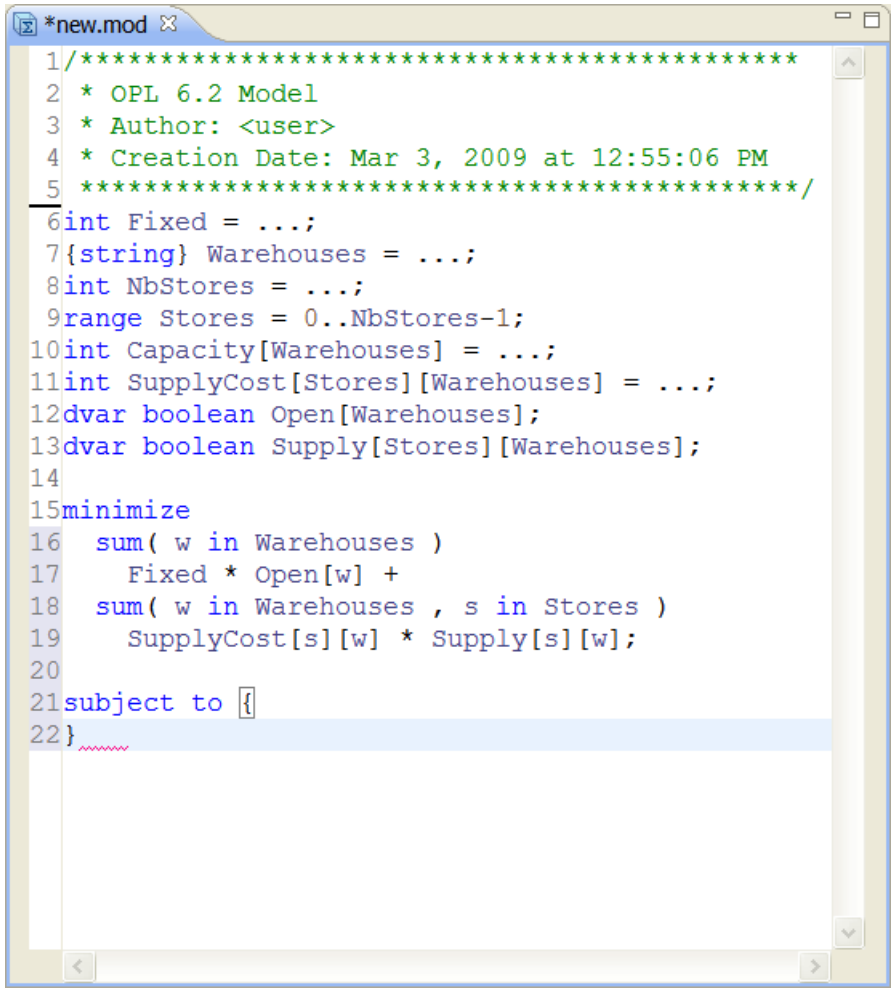
   The popup window will open, displaying entries that are appropriate for the context of your current cursor location in the file.

2. **To filter the display**, type one or more characters of the keyword or function you are searching for, before or after pressing **Ctrl + space**. For example, here the user has typed the letters **su** and then selected the template **Subject to block**.

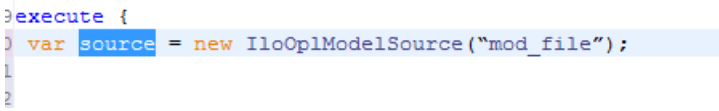


Note that when you select a template in the Model Completion popup window a preview window is displayed to the right that allows you to see the contents of the template before you select it.

3. **To insert the template**, double-click it or select it and press **Enter**. In this example, the user has selected the **Subject to block** template, and it is inserted into the editor at the cursor position:



4. **If the template contains variables**, your cursor position is automatically shifted after insertion to highlight the first variable, so that you can replace the placeholder variable with the real one.



   If the template code contains multiple variables, you can navigate to them by pressing the **Tab** key.

You can continue in this fashion as you write your model, pressing **Ctrl + space** to see a list of possible keywords, functions, and templates at any point.

Editing or creating Model Templates

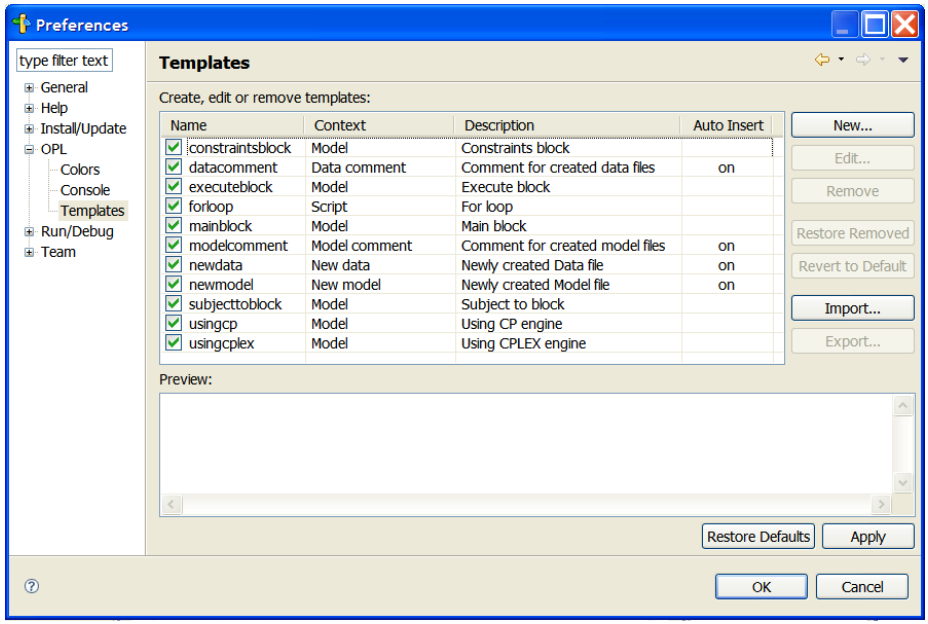
Model Templates can be edited or created using the OPL Preferences editor. A number of templates are supplied with OPL, but you can edit them to suit your purposes, or create your own.

1. **To open the OPL Preferences window**, select **Window > Preferences** from the main menu.

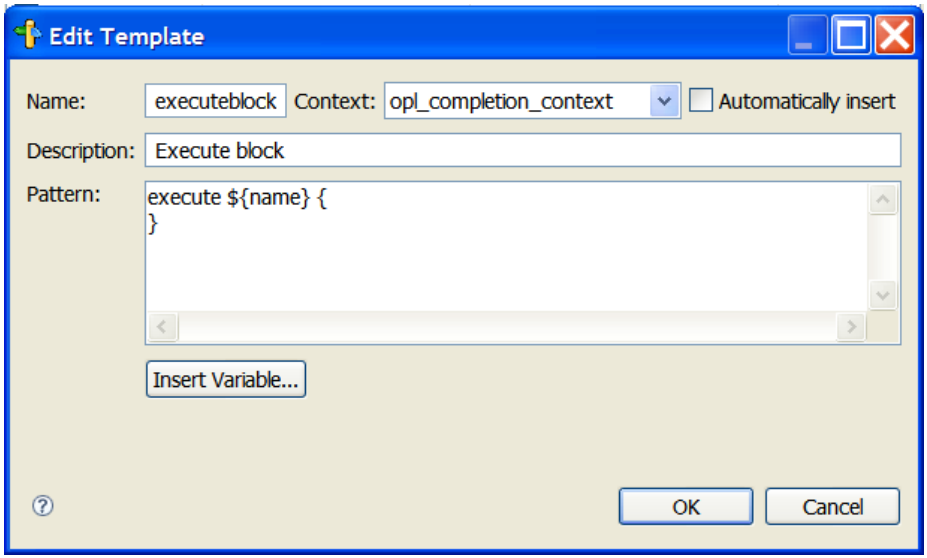
   The Preferences window is displayed.

2. **To display the Templates editor**, in the left panel, expand the **OPL** item and select **Templates**.

   Details for the Templates are displayed:



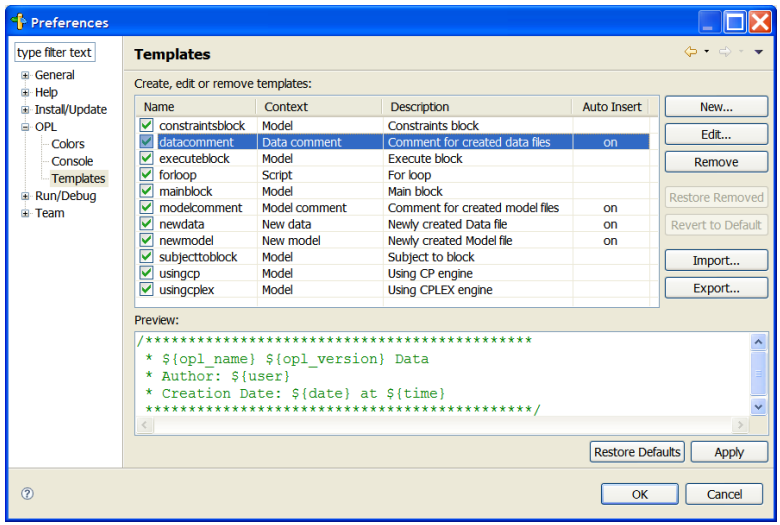
3. **To edit an existing template**, select it and press the **Edit** button.



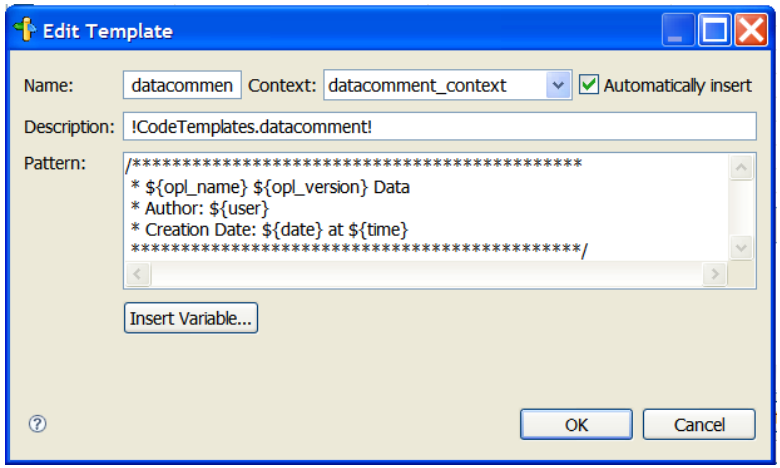
In the example above, the user is editing the **Execute block** template. (Additional instructions about the controls on this window are contained in the following steps on creating a new template.)

Note: These screenshots show a sample of templates, not necessarily the full set shipped with OPL.

Note that not all of the templates shown will display in the Model Completion popup window. The reason for this is that some of the default templates are used for other purposes in OPL, such as inserting default headings when you create a new data or model file. For example, the following screenshot shows the **datacomment** template selected, with its content in the **Preview** area:

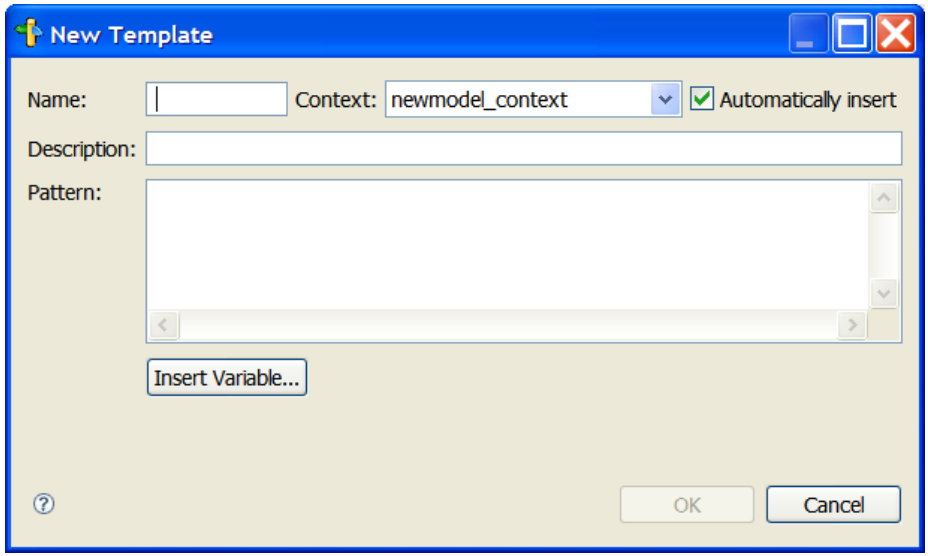


If you edit this template (by clicking the **Edit** button), you can see that it has been defined using the **Automatically insert** option, and that the template content contains several variables, which are read from system or OPL variables and filled in when the user creates a new data file.



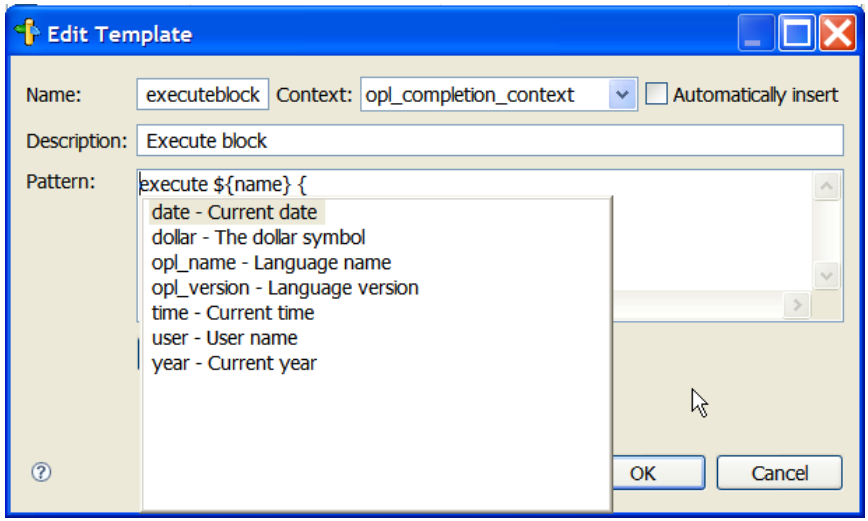
All templates supplied with OPL can be edited, so if you want to change the headings in the **datacomment** or **modelcomment** templates to comply with your company's standards, you can.

4. **To create a new template**, click the **New** button to display the New Template window:



Once the New Template window is open, use the following steps to create a new template of your own:

- Type a name for your template in the **Name** field.
- Type a context for your template in the **Context** field. Available context types are:.
  - **opl_completion_context** — used when creating OPL code templates
  - **script_completion_context** — used when creating OPL Script code templates
- In most cases, you will not use the **Automatically insert** checkbox. This is used when creating templates that are automatically inserted into new files as they are created, such as the supplied **datacomment** and **modelcomment** templates.
- Type a description for your template in the **Description** field.
- Type the code for your template in the **Pattern** field:
  - Type the code as you would normally, in the editor itself.
  - To insert line breaks, press **Enter**.
  - To insert OPL or system variables into your code (such as **User name**, **Current date**, etc.), click the **Insert Variable** button and double-click the variable or variables you want to insert.



- When you have finished entering your template, press **OK** to save it and return to the OPL Preferences window.

5. **To export a template in XML format** (for example, for sharing with other OPL developers), select the template and click the **Export** button.

   In the popup window, choose a directory for the exported XML, enter a **File name**, and click the **Save** button.

6. **To import a template in XML format** (for example, when receiving a template shared by another OPL developer), click the **Import** button.

   In the popup window, navigate to the directory containing the exported XML, select the template you want to import, and click the **Open** button.

Local History and its related features

OPL provides a limited form of version control called **Local History** that allows you to track and compare different versions of your files as you edit them over the life span of a project.

For example, if you edit the same model file several times, all versions of the file are still available to you. You can use the **Compare With** and **Replace With** commands to compare different versions of a file or revert to previous versions of a file or its contents.

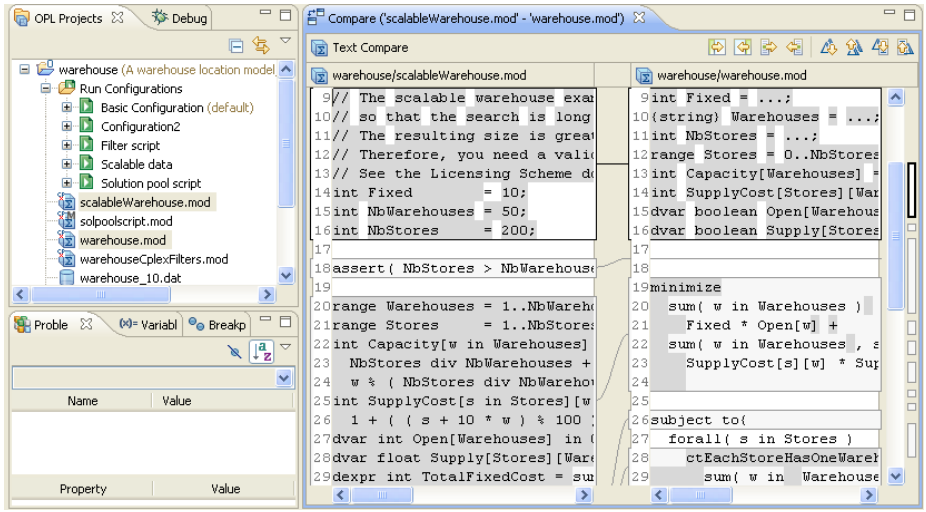
Each of these features is described in the following sections

'Compare With' features

Compare With Each Other

To compare two files in the same project with each other:

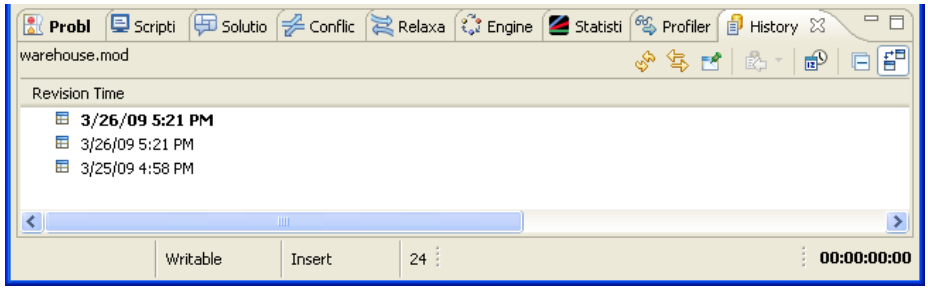
1. In the OPL Projects Navigator, highlight the two files you want to compare and right-click to display a context menu and choose **Compare With>Each Other** from that menu.
2. The files are opened in the Editing Area in a special view that allows you to view them side by side, with the differences between the two files highlighted:



Compare With Local History

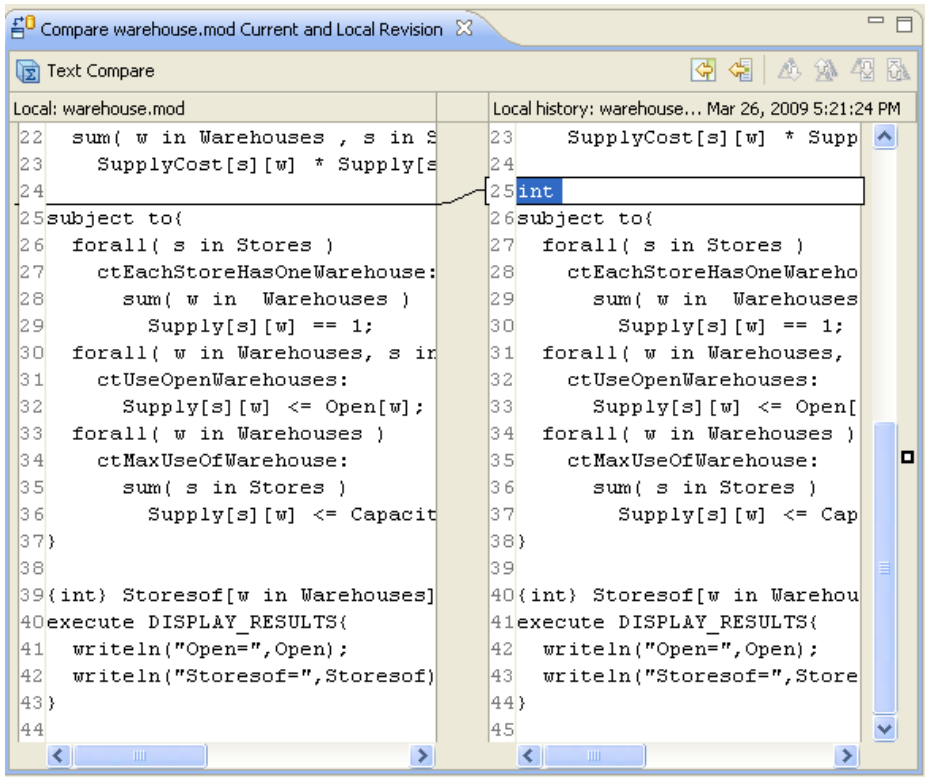
To compare a file with another version of itself in Local History:

1. In the OPL Projects Navigator, highlight the file you want to compare to its own Local History versions and right-click to display a context menu. Then choose **Compare With>Local History** from that menu.
2. In the Output Area of the OPL IDE, a list appears of the different versions of this file in Local History:



3. Double-click on the version of the file that you want to compare to the current version.

   The files are opened in the Editing Area in a special view that allows you to view them side by side, with the differences between the two files highlighted:

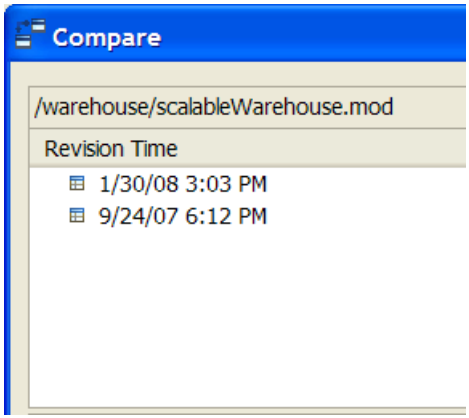


'Replace With' features

Replace With Local History

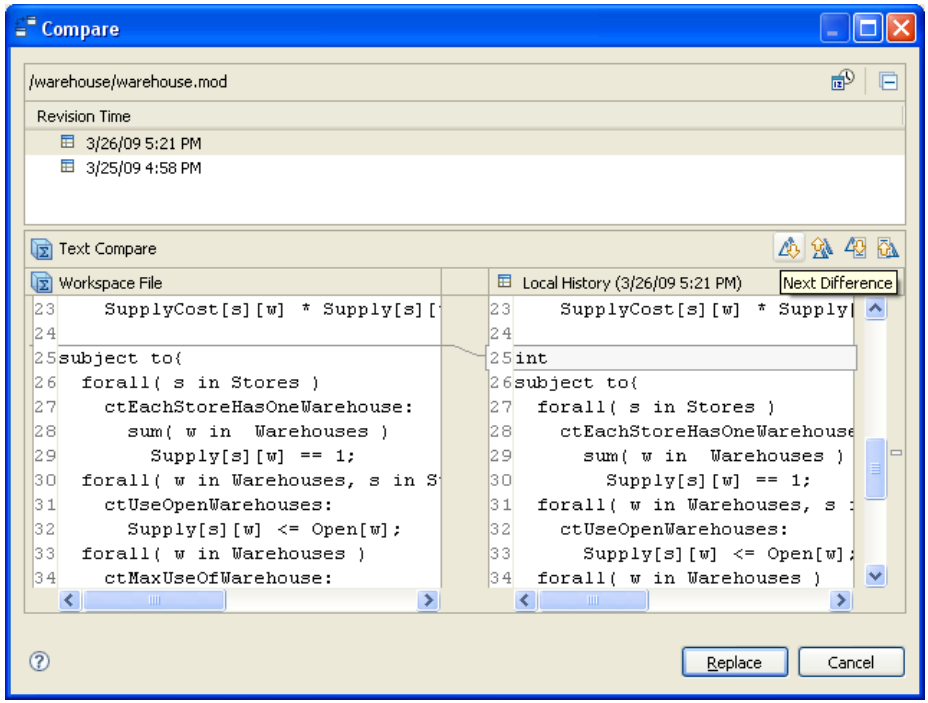
To replace a file with a selected version of itself from Local History:

1. In the OPL Projects Navigator, highlight the file you want to revert to one of its Local History versions and right-click to display a context menu. Then choose **Replace With>Local History** from that menu.
2. A popup window appears in which you can view the different versions of this file in Local History:



3. Double-click on the version of the file that you want to compare to the current version and use as a candidate for replacement.

   The files are opened in a special view that allows you to view them side by side, with the differences between the two files highlighted:



You can use the icons that appear on the right of the lower window to move to the next and previous differences in the files, and thus determine whether you want to replace the contents of the current file with the contents of the version you are comparing it to.

Replace With Previous Version

To replace a file with its previous version in Local History:

1. In the OPL Projects Navigator, highlight the file you want to revert to its previous version and right-click to display a context menu. Choose **Replace With>Previous Version** from that menu.
2. The file is automatically replaced with the version of the file from Local History that immediately precedes it. This creates a new version in Local History.

Menu commands and equivalent toolbar buttons

Lists the available OPL commands menu by menu.

In this section

General description
Provides an overview of the menu commands in the OPL IDE.

File menu
Describes the OPL options available on the File menu in the IDE.

Edit menu
Describes the OPL options available on the Edit menu in the IDE.

Navigate menu
Describes the OPL options available on the Navigate menu in the IDE.

Run menu
Describes the OPL options available on the Run menu in the IDE.

Window menu
Describes the OPL options available on the Window menu in the IDE.

Help menu
Describes the OPL options available on the Help menu in the IDE.

General description

Use the menus in the menu bar to choose commands, such as **File>Save**, and to display dialog boxes to perform various tasks.

Certain menu commands have their own buttons in one of the toolbars just below the menu bar (the button label in the tooltip may be slightly different from the command name) or on the toolbar of a specific view.

Some menu commands have a keyboard shortcut, indicated in the right-hand column of the menu. For example, **Dynamic Help** has the shortcut **F1**, which means that you can obtain help on an OPL keyword in the Editor by selecting the keyword and then either clicking the item **Help>Dynamic Help** or pressing the key **F1**.

Many of the menu commands are standard or self-explanatory. The following sections contain tables that list menu commands and, when available, equivalent toolbar buttons and keyboard shortcuts, and provide a description of each command.

File menu

The OPL commands of the File menu are listed below. Some menu items may also have toolbar buttons or keyboard shortcuts associated with them.

File menu commands

Command	Submenu Commands, if any	Shortcut	Description
File			
New	**OPL Project** **Project** **Run Configuration** **Model** **Data** **Settings** **Folder** **File** **Untitled Text File** **Example** **Other**		Displays a submenu to specify a model, project, data, settings file, or other OPL resource. If you choose **OPL Project**, the IDE opens the New Project dialog box. If you choose **Example**, the IDE opens the New Example wizard. If you choose other items, the IDE displays a dialog box or wizard appropriate to each type.
Import	**Example** **Existing OPL 6.x projects** **Migrate OPL 5.x projects**		If you choose **Example**, the IDE opens the New Example wizard. Displays a submenu to specify which type of project you want to import: If you choose **Existing OPL 6.x projects**, opens the Import Projects dialog box. This is for opening projects created in OPL 6.x.

Command	Submenu Commands, if any	Shortcut	Description
			If you choose **Migrate OPL 5.x projects**, the IDE opens the Import OPL 5.x Projects dialog box. This is for migrating projects created in OPL 5.0 or higher.
Copy Files to Project			Opens a dialog box so that you can search for files to import and insert into existing projects in the OPL Projects Navigator.
Open File in Editor			Opens an Open File dialog box requesting the file name you want to open in the Editing Area.
Close		**Ctrl+W**	Closes the active file in the Editing Area. If the file is unsaved, you are prompted to save.
Close All		**Ctrl+Shift+W**	Closes all of the active files in the Editing Area. If the files are unsaved, you are prompted to save.
Save		**Ctrl+S**	Saves the current file in the Editing Area with its current name and at the current location. Note that executing a run configuration automatically saves its files.
Save As			Saves the current edited file with a new name and/or location.
Save All		**Ctrl+Shift+S**	Saves all the files in the Editing Area with their current names and at the current location.
Print		**Ctrl+P**	Prints the current document.
Exit			Exits OPL Studio and prompts you to save any unsaved project.

Edit menu

The OPL commands of the Edit menu are listed below. Some menu items may also have toolbar buttons or keyboard shortcuts associated with them.

Edit menu commands

Command	Submenu Commands, if any	Shortcut	Description
Edit			Many items in this menu are inactive unless a document is being edited in the Editing Area.
Undo		**Ctrl+Z**	Undoes the last action in the current Editing Area (unlimited).
Redo		**Ctrl+Y**	Redoes the previously undone action in the current Editing Area (unlimited).
Cut		**Ctrl+X**	Removes the selected text and copies it to the clipboard, so you can paste it elsewhere.
Copy		**Ctrl+C**	Copies the selected text from the Editing Area or Output Area to the clipboard, so you can paste it elsewhere.
Paste		**Ctrl+V**	Pastes from the clipboard to the current Editing Area.
Delete		**Delete**	Deletes the currently-selected object.
Find/Replace		**Ctrl+F**	Displays the Find dialog for specifying search criteria. Also allows you to replace specified strings.
Find Next		**Ctrl+K**	Finds the next occurrence of the text displayed in the Find dialog.
Find Previous		**Ctrl+Shift+K**	Finds the previous occurrence of the text displayed in the Find dialog.
Incremental Find Next		**Ctrl+J**	Finds the incremental next occurrence of text that you type. For example, press **Ctrl+J** and start to type `dvar`, and you find the next occurrence of the string as you type. While in this mode, the up and down cursor keys can be used to navigate between matches, and the search can be canceled by pressing the left or right cursor keys, the **Enter** key, or the **Esc** key.

Command	Submenu Commands, if any	Shortcut	Description
			This text is added in the background to the Find dialog, which can be displayed by pressing **Ctrl+F**.
Incremental Find Previous		**Ctrl+Shift+J**	Finds the incremental previous occurrence of text that you type. For example, press **Ctrl+Shift+J** and start to type `dvar`, and you find the previous occurrence of the string as you type. While in this mode, the up and down cursor keys can be used to navigate between matches, and the search can be canceled by pressing the left or right cursor keys, the **Enter** key, or the **Esc** key. This text is added in the background to the Find dialog, which can be displayed by pressing **Ctrl+F**.
Toggle Comment		**Ctrl+/**	Transforms the line the cursor is current in to comments, using the ***//*** delimiter. Or, if the line has been previously commented, deletes the comment delimiter. This works only in model files at present. You can also add comments manually using ***/**** and *****/*** as opening and closing delimiters. However, in this case, using the **Toggle Comment** command to remove the comments is ineffective.
Word Completion		**Alt+/**	Searches upward for a similar string in the same file and completes with the first string that matches the same beginning letters. The search begins upward at the left of the current cursor position.

Navigate menu

The OPL commands of the Navigate menu are listed below. Some menu items may also have toolbar buttons or keyboard shortcuts associated with them.

Navigate menu commands

Command	Submenu Commands, if any	Shortcut	Description
Navigate			
Go to Line		**Ctrl+L**	Displays the Go To dialog box for specifying a line where the cursor should be placed in the Editing Area.

Run menu

The OPL commands of the Run menu are listed below. Some menu items may also have toolbar buttons or keyboard shortcuts associated with them.

Run menu commands

Command	Submenu Commands, if any	Shortcut	Description
Run			
Run		**Ctrl+F11**	Submits the last-executed run configuration for execution. After you click **Run**, the **Pause** button becomes available, as well as other buttons of the execution toolbar in certain cases. You can click the arrow next to the button and select the file to debug from the contextual menu. Note that executing a run configuration automatically saves its files.
Debug		**F11**	Starts the execution of the active run configuration and stops the solving engine at each breakpoint. You can click the arrow next to the button and select the file to debug from the contextual menu.
Browse			Builds or rebuilds the model tree of the data structures defined in the active model or project to allow browsing the project in the Problem browser without solving it.

Command	Submenu Commands, if any	Shortcut	Description
			You can click the arrow next to the button and select the file to browse from the contextual menu.
Run History	**<list of recently-run run configurations**		Displays a selectable list of recently run or debugged files.
Run As	**<list of user-defined run configurations>**		Displays a selectable list of user-defined run configurations.
Open Run Dialog			Opens a dialog to define a list of run configurations to run in standard run mode.
Debug History	**<list of recently-run run configurations**		Displays a selectable list of recently run or debugged files.
Debug As	**<list of user-defined run configurations>**		Displays a selectable list of user-defined run configurations.
Open Debug Dialog			Opens a dialog to define a list of run configurations to run in debug mode.
Browse History	**<list of recently-run run configurations**		Displays a selectable list of recently run or debugged files.
Browse As	**<list of user-defined run configurations>**		Displays a selectable list of user-defined run configurations.
Browse Configurations			Opens a dialog to define a list of run configurations to run in browse mode.
Export external data			Only active after a run configuration has been executed and completed. Opens a dialog to choose a filename and a location for the exported external data.
Export internal data			Only active after a run configuration has been executed and completed. Opens a dialog to choose a filename and a location for the exported internal data.
External Tools	**Run As** **Open External Tools Dialog** **Organize Favorites**		**Run As** displays a selectable list of user-defined run configurations. Opens a dialog to define a list of run configurations to run in the background in the Script log tab.

Command	Submenu Commands, if any	Shortcut	Description
			Opens a dialog to organize the predefined run configurations.

Window menu

The OPL commands of the Window menu are listed below. Some menu items may also have toolbar buttons or keyboard shortcuts associated with them.

Window menu: commands and equivalent buttons

Command	Submenu Commands, if any	Shortcut	Description
Window			
Show View	**Conflicts** **Engine log** **OPL Projects** **Problem browser** **Profiler** **Relaxations** **Script log** **Solutions** **Statistics** **Other**		Choosing one of the view names in the list displays that view in the OPL IDE. Choosing **Other** displays a dialog of additional views.
Preferences			Opens the Preferences dialog.

Help menu

The OPL commands of the Help menu are listed below. Some menu items may also have toolbar buttons or keyboard shortcuts associated with them.

Help menu: commands and equivalent buttons

Command	Submenu Commands, if any	Shortcut	Description
Help			
Welcome			Display the Welcome window in the OPL IDE.
Help Contents			Launch the documentation in its own viewer.
Dynamic Help		**F1**	Display the Help window in the OPL IDE. If a keyword is selected, display help for that keyword.
Key Assist		**Ctrl+Shift+L**	Display a popup list of keyboard shortcuts.
About IBM ILOG OPL IDE			Displays a window that indicates the version of OPL and the products used by this version of IBM ILOG OPL. Also contains copyright information.

Right-click context menu commands

Describes the OPL commands available on context menus when you right-click a project, model, data, settings file, or a run configuration, and when you right-click in the Problem browser or in the Output Area.

In this section

For projects, models, and data
Describes the right-click menus available for projects, models, data, settings in the OPL Projects Navigator.

For run configurations
Summarizes the right-click OPL commands available from run configurations.

In the Output Area
Describes the right-click OPL commands available from the tabs of the Output Area.

For projects, models, and data

Right-click context menus are available in the OPL Projects Navigator on projects, model, data, and settings files.

Projects

OPL Projects Navigator

By just right-clicking the background of the OPL Projects Navigator, you can create new projects or project resources, and you can import/migrate existing projects. See *Creating a project* in *Getting Started with the IDE* for more information on how to create projects.

See the *OPL 6.x Migration Guide* for more information on how to import projects created in previous versions of OPL or open OPL 6.x projects.

Project Files

Most commands available when you right-click a project are equivalent to buttons on the OPL Projects Navigator toolbar or to commands in the main menus.

Model, data, and settings files

Two different sets of actions are possible depending on whether you right-click the file name of a model file, data file, or settings file in the project or within a run configuration in the OPL Projects Navigator:

For run configurations

OPL commands are available in the context menu by right-clicking on a run configuration in the OPL Projects Navigator. See *The Run options* section of *Introduction to the OPL IDE* and *Executing a project* in *Getting Started with the IDE* for more information on how to work with run configurations.

In the Output Area

Right-click commands available in the different tabs of the Output Area vary by tab.

Preferences and options

A general presentation of the Preferences dialog box and of the settings editor for OPL, CP, and MP options.

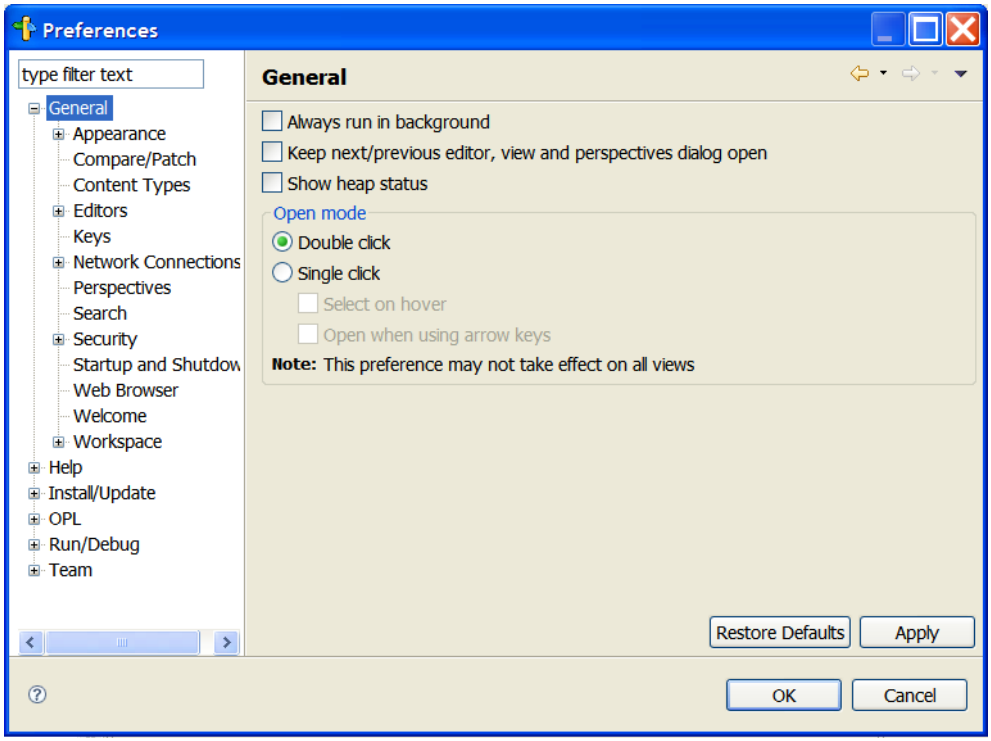
In this section

Setting IDE preferences
Presents the Preferences dialog box and its organization.

Setting programming options
Presents the settings editor which opens when you double-click a settings (`.ops`) file.

Setting IDE preferences

Choose **Window>Preferences** to open the dialog box shown in *Preferences, user interface start options*.



Preferences, user interface start options

The options are organized in pages. To access a page, click a tree item on the left. You can expand or collapse the option tree.

The default values are displayed initially. When you change the default value of a preference, a red exclamation mark appears next to the changed setting.

To set your preferences, click **Apply** or **OK**, and the IDE will save the new value. The changed options are restored the next time you start the IDE.

To reset the default values, click **Restore defaults**.

See also *Changing an MP option value* in *Getting Started with the IDE*.

See *IDE Preferences* for a description of each preference.

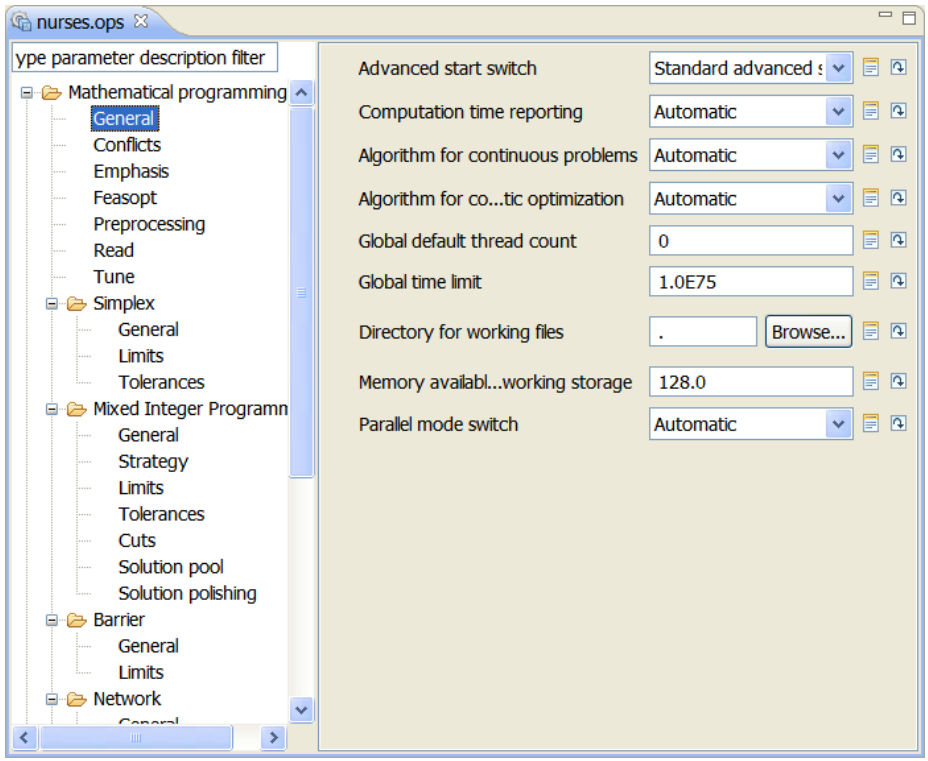
Setting programming options

Unlike the IDE Preferences, which apply to an IDE session and are saved with it, MP, CP, and language output options are set within an `.ops` file associated with a project and run configuration. When you double-click an `.ops` file in a project or run configuration, the settings editor opens (see *Settings editor*).

To access the option pages:

1. Make sure the project is selected.
2. Create a settings file as explained in *Adding a settings file* in *Getting Started with the IDE*.

   In the Editing Area, you see the settings editor.
3. Use the settings editor as explained below.



Settings editor

Mouseover the first icon to the right of each field to see a short description of it and its parameters.

Click the second icon to the right of the field to reset it to its default value.

For a description of individual options, see:

- OPL language options
- Constraint programming options
- Mathematical programming options

Icons

Describes the icons used in the IDE to represent types in the Problem browser, and types and sets in call stacks.

In this section

Types in the Problem browser
Icons used in the Problem browser.

Types and sets in call stacks
Lists the types of icons used in call stacks.

Types in the Problem browser

Icons for types in the Problem browser

Type	Icon	Type	Icons
Data section (or Postprocessing Data section)		Decision Variables section	
Constraints section		Constraint	
Integer element		Integer set	
Integer array (data, dvar, dexpr, or constraint)		Float element	
Float set		Float array (data, dvar, dexpr, or constraint)	
Tuple		Tuple set	
Tuple array		Range	

Types and sets in call stacks

Icons for types in the call stack

Type	Icon	Type	Icon
Array		Boolean	
Date		Frame	
Null		Number	
Other		Proxy	
Scope		Stack	
String		This	
Undefined			

Identical icons are used for sets, except that they are enclosed in a pair of curly brackets {}, for example  for a set of strings.

Processes and Procedures

Provides more detail on the execution process, on the Problem Browser, and on project management.

In this section

What happens when you execute a run configuration
Describes the internal execution process and provides details of each step.

Doing more with the Problem Browser
Describes all you can do with the Problem Browser beyond examining the structure of a model after execution.

Doing more with projects
Explains how to open and reuse existing files, work with several projects, use templates, consider the order of files in a project, and save and close projects.

Generating output files
Describes how the IDE enables you to generate model and data as output files that you can later use from a command prompt and with OPL interface libraries.

Launching OPL run configurations in the background
Explains the two methods of launching OPL run configurations from the IDE as a background process, and the options available when doing this.

What happens when you execute a run configuration

When you trigger the execution of a run configuration, the IDE analyses the model and checks it for syntactic errors, builds representations, solves the problem itself by finding values, and displays some results.

The internal process of execution

Once a project is open in the IDE, you execute it by right-clicking the run configuration you want to run and choosing **Run > <run_configuration_name>** from the context menu in OPL Projects Navigator, as explained in the *The Run options* section of *Introduction to the OPL IDE*

Before the IDE begins executing a run configuration, it compiles the OPL statements into an internal representation that is better suited for execution. An OPL statement must be correct before the IDE can execute it.

- If the model contains errors displayed in the **Problems** tab, the IDE does not start the solving engine. You must correct any errors as indicated in *Dealing with errors* in *Getting Started with the IDE* before running the model. When you have corrected the errors, launch the run configuration again.
- When the model syntax is correct, the IDE starts to execute the model.

Note: Executing a run configuration automatically saves unsaved files.

When the IDE starts to execute the run configuration, you can use the buttons in the execution toolbar to **Pause** or **Abort** the run. See *The execution toolbar* for details.

The functions in the toolbar in **Debug** view are not available unless you launched the run in debugging mode using the Debug button  .

Summary of the execution process

When the OPL Studio executes a run configuration, it typically does the following:

1. analyzes the model,
2. checks for syntax and semantic errors in the model,
3. builds the tree representing the data structures of the model in the Problem Browser,
4. executes the model if it has no errors,
5. displays run status messages in the Status Bar and changes the status indicator (see *The Status Bar* for more information), and
6. updates the tabs of the Output Area as execution proceeds.

Launching a selected run configuration triggers a chain of events. With a simple model, extraction, execution, and solution display are nearly simultaneous.

Details of the execution process

1. **Analysis of the model**

   The IDE first analyzes the model and fills in the Problem Browser with summary information about the data structures defined in the model and with the list of solutions. See *Understanding the Problem Browser* in *Getting Started with the IDE* for more information.

2. **Syntax and semantic checks**

   If the IDE finds errors that were not detected dynamically when you wrote the model, it stops the run so that you can correct the errors. See *Dealing with errors* in *Getting Started with the IDE*.

3. **Execution of the model**

   If there are no errors, the IDE extracts the model and starts execution. A message appears in the Status Bar and the status indicator animates to indicate the solve process is running. If intermediate solutions are found before the final solution, they appear in the **Solutions** output panel.

4. **Output Area**

   When the IDE finds the optimal solution, the solve operation stops. The tab names for the panels of the Output Area that have received content are highlighted. They revert to normal display as soon as you click them to see their content.

   - **Problems**: status and error messages
   - **Scripting log**: execution of flow control, preprocessing, postprocessing IBM ILOG Script blocks if the model contains any.
   - **Solutions**: feasible and final
   - **Conflicts**: if the CPLEX model is infeasible
   - **Relaxations**: if the CPLEX model required relaxations to make it feasible
   - **Engine log**: execution history in text mode
   - **Statistics**: table and progress chart
   - **Profiler**: performance (speed and memory consumption)

Results

When executing a run configuration, the IDE uses the solving engine to compute one or more solutions. If the project contains an MP model (solved by the CPLEX® engine), the IDE detects infeasibility and conflicts, if any, and displays the results in the various tabs of the Output Area, as explained in *The Output tabs* in *Getting Started with the IDE*.

Note: There is no support for conflicts and relaxations for models solved by the CP Optimizer engine.

Doing more with the Problem Browser

Describes all you can do with the Problem Browser beyond examining the structure of a model after execution.

In this section

Introduction
A summary of what you can use the Problem Browser for.

Browsing a model without execution
Describes how to browse a model without executing it if you are not interested in the solution.

Element usage in browsing vs. solving
Describes model elements that are used differently when browsing than when solving.

Navigating the model file
Explains how the Problem Browser helps you find model elements.

Postprocessing
Describes how the model browser displays the results of postprocessing.

Displaying individual table views
Describes how to display table views of values.

Resulting views
Describes table views for data and for variables.

Introduction

You can use the Problem Browser to:

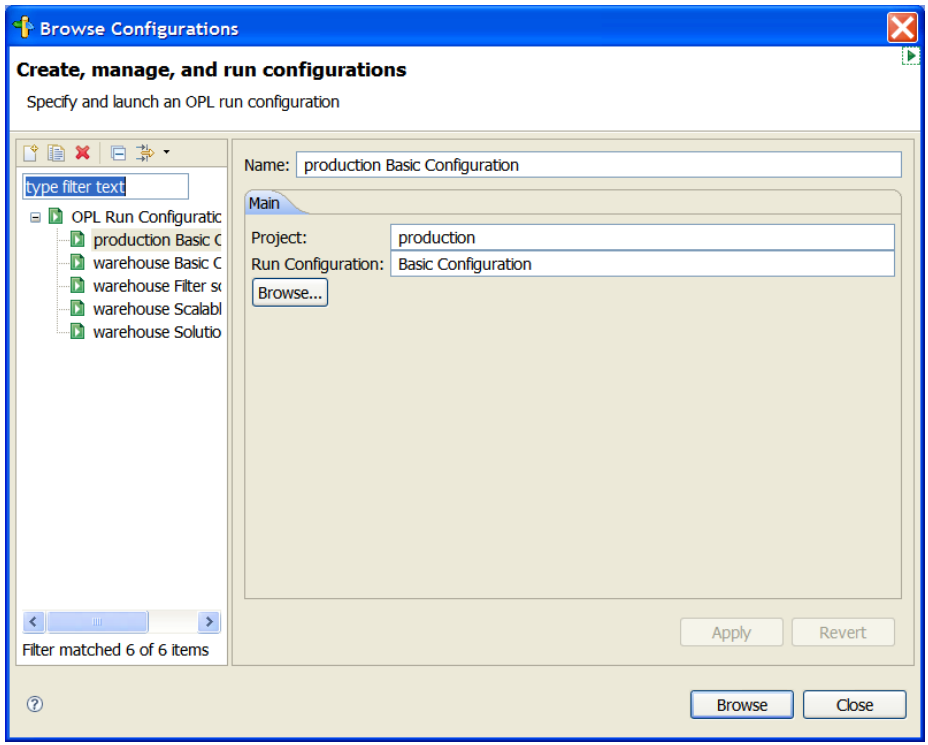
- examine the structure of the model without executing it: see *Browsing a model without execution*
- navigate through the model file displayed in the Editing Area: see *Navigating the model file*
- examine the solutions to the problem and display additional views of them after executing the model: see *Displaying individual table views*
- work with solution pools: see *Working with the solution pool* in *IDE Tutorials*.

Browsing a model without execution

If, at some point, you want to examine the structure of a model but are not interested in the solution, you can browse the model without executing it.

To browse the model before execution:

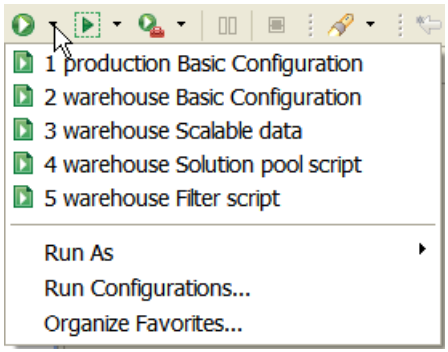
1. Use the **File>New>Example** menu command to open the **production** example.
2. To browse the model, select `production.mod`, then click the arrow of the **Browse** button  in the execution toolbar and choose **Browse Configurations**.
3. The Browse dialog window opens:



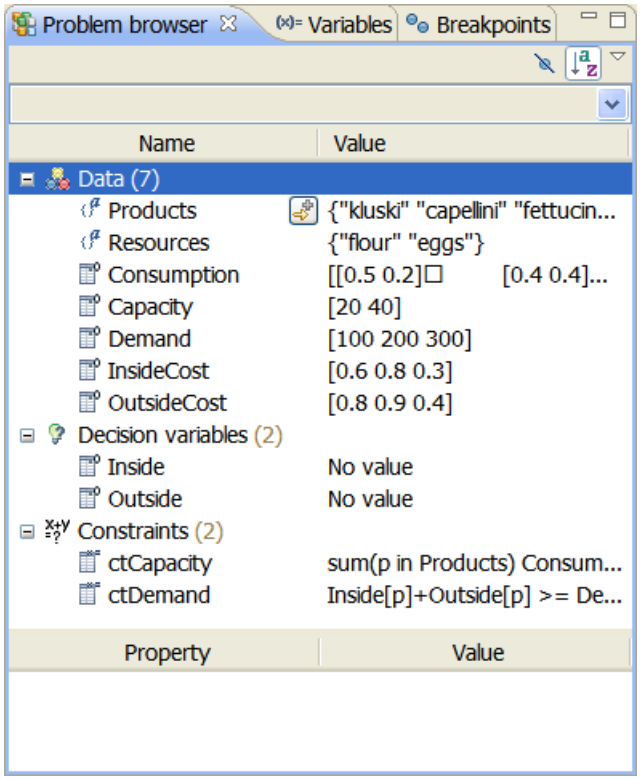
   Enter the information as shown in the screenshot and click the **Apply** button.

4. The run configuration is added to the left part of the Browse dialog. Now you can click the **Browse** button on this same dialog to launch this run configuration in browse mode.

Note: You could also close the Browse dialog window and then click the arrow of the **Browse** button  and choose the run configuration you just created from the list:



Compare *Browsing a model without execution (`production.mod`)* with *Problem Browser after execution (`product.mod`)* in *Getting Started with the IDE* (or run the same run configuration in run mode).



Browsing a model without execution (`production.mod`)

Notice that no solutions are listed and no values are available for decision variables before execution.

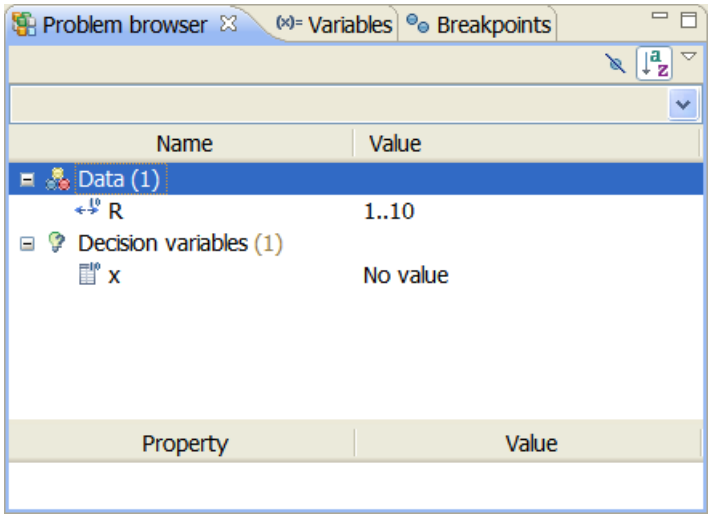
Element usage in browsing vs. solving

Because the “browse” action displays the structure of the problem expressed by the model, it does not take into account whether the model elements are used or not at execution time and shows them all. In contrast, a “solve” action is interested by default only in the model elements that are part of the solutions (unless you check **Force element usage**, see OPL language options in *IDE Reference*). This is why model elements that are not used by the solving algorithm are not visible in the Problem Browser whereas they are if you just click Browser before execution. For example, if you browse the model shown in *dexpr arrays: browsing versus solving*, the `dexpr` array is visible in the Problem Browser. If you execute this model, `y` does not belong to the solution because it does not comply with the `allDifferent` constraint. Therefore, the **Decision expressions** line of the Problem Browser remains empty.

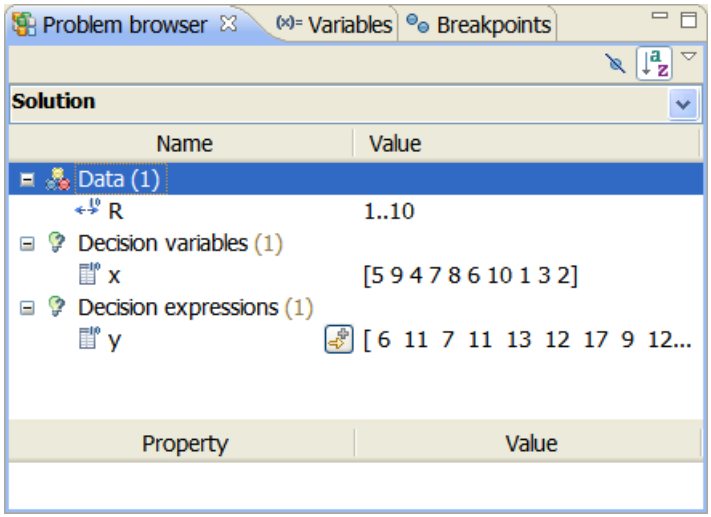
dexpr arrays: browsing versus solving

```
using CP;

range R = 1..10;

dvar int x[R] in R;

dexpr int y[i in R] = x[i] +i;

subject to {
   allDifferent(x);
}


execute {
   writeln(x);
   writeln(y);
}
```

Compare *Problem Browser: dexpr after solve* and *First occurrence of entry selected in Problem Browser*.



Problem Browser: no dexpr after browse



Problem Browser: dexpr after solve

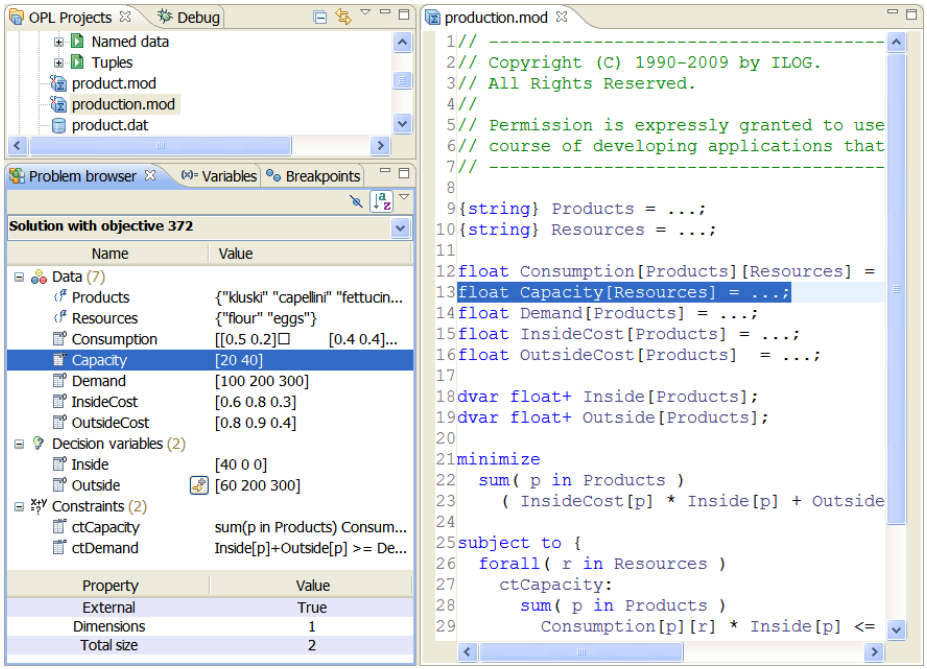
Navigating the model file

If you click one of the elements in the Problem Browser, the IDE selects the line in the model that contains the first occurrence of that element, for example,

To navigate the model using the Problem Browser:

1. After running the default run configuration of the `production` example, click the line that contains the **Capacity** element.

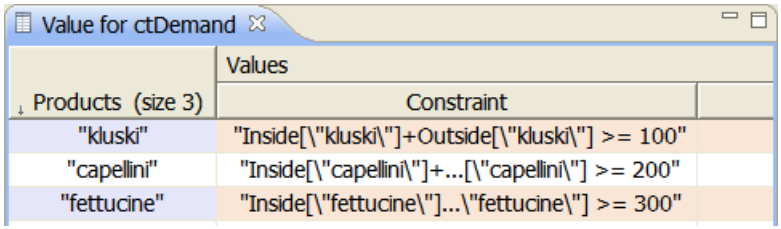
   The first occurrence of **Capacity** (its declaration) is highlighted in the open **production.mod** model file in the Editing Area, and the element properties appear in the Property table of the Problem Browser, as shown in *First occurrence of entry selected in Problem Browser*.



First occurrence of entry selected in Problem Browser

2. Double-click the **ctDemand** constraint to display the details in a new window in the Editing Area.

   The `forall` iterations appear in a tree view that allows you to expand or hide the details of each iteration. as shown in *Expanded constraints (`production.mod`)*.



Expanded constraints (`production.mod`)

Postprocessing

To see how the Problem Browser displays postprocessing results, open the `mulprod` example, which is available at:

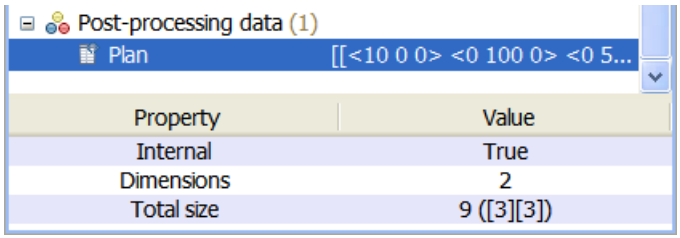
`<OPL_dir>\examples\opl\mulprod`

where `<OPL_dir>` is your installation directory.

When you run the default run configuration for this model, under the **Postprocessing** item of the Problem Browser, you can see that this line of the model

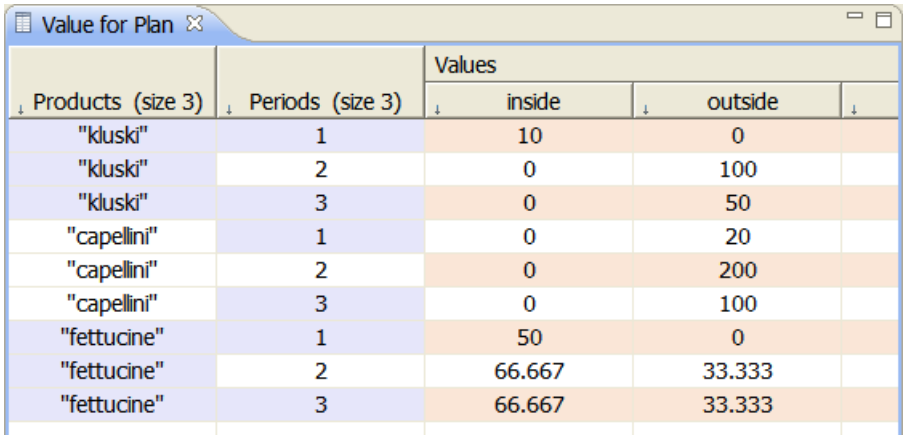
```
plan Plan[p in Products][t in Periods] = <Inside[p,t],Outside[p,t],Inv[p,t]>;
```

appears as shown in *Postprocessing in Problem Browser (`mulprod` example)*.



Postprocessing in Problem Browser (`mulprod` example)

If you double-click on the Plan element in the Problem Browser, detail for it is shown in a new window in the Editing Area:



Postprocessing detail in the Editing Area (`mulprod` example)

You can see that the window is divided in two parts, one for the indexers, and one for the values of the arrays. The array content is displayed "flattened," with a number of rows that is exactly the total number of items.

Displaying individual table views

From the Problem Browser, you can also display table views for individual elements.

1. Open the `production` example in the OPL IDE.
2. Run the **Tuples** run configuration.
3. Select an element in the Problem Browser, for example, **Inside** and click the **Show data view**  button that appears.

   The IDE displays a view for that entry. The presentation of the element's content depends on the type of object displayed.

Resulting views

Data views

If you double-click **Capacity** under **Data**, the IDE displays details of the data, as shown in *Table view of Capacity data (`product.mod`)*.

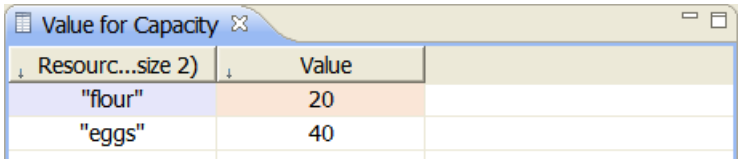


Table view of Capacity data (`product.mod`)

Variables views

There are similar views of Variables which represent the results of the problem. Sometimes you may want to display the Data views and Variables views together as they have a similar appearance and can be compared easily.

For example, if you double-click **Inside** under **Decision variables**, you can see the solution details for these variables in *Table view of the Inside variable (`product.mod`)*.

Value for Inside

Products (size 3)	Periods (size 3)	Values		
		Value	Reduced cost	Sensitivity range
"kluski"	1	10	0	[-∞..10] [10..∞]
"kluski"	2	0	0	[-∞..0] [0..∞]
"kluski"	3	0	0	[-∞..0] [0..∞]
"capellini"	1	0	0	[-∞..0] [0..∞]
"capellini"	2	0	0.05000000000000004	[-66.6666666...66..0] [0..∞]
"capellini"	3	0	0.05000000000000004	[-66.6666666...66..0] [0..∞]
"fettucine"	1	50	0	[-∞..50] [50..∞]
"fettucine"	2	66.667	0	[-∞..66.666...66666667..∞]
"fettucine"	3	66.667	0	[-∞..66.666...66666667..∞]

Table view of the Inside variable (`product.mod`)

Doing more with projects

Explains how to open and reuse existing files, work with several projects, use templates, consider the order of files in a project, and save and close projects.

In this section

Reusing existing files and projects
Describes the various ways of opening existing projects and reusing them in other projects.

Working with several projects
Describes how to work with multiple projects.

Using templates
Describes how to use the default templates.

Order of files
How the order of data and settings files affects execution results and how to determine that order.

Saving a project
Describes the various ways of saving projects and their effects.

Closing projects
Describes the various ways of closing projects and their effects.

Reusing existing files and projects

In some cases, it is quicker to create an OPL project by opening and editing an existing one. As a distributed product, IBM® ILOG® OPL comes with many code samples and industry models which you can reuse as a starting point for your own projects. The Language and Interfaces Examples manual provides a detailed description of the optimization issues they express, as well as a quick access to the files.

The following procedures suggest how you can open and reuse the examples in the OPL distribution, or your own existing projects.

To reuse an existing project, first you should open and copy it:

1. From the OPL main menu, choose **File>Import>Existing OPL 6.x projects**, or right-click in the OPL Projects Navigator and choose **Import>Existing OPL 6.x projects** from the context menu.
2. To select one of the distributed examples for copying and reuse, navigate to the folder under `<OPL_dir>\examples` that contains the model you want to reuse, and open it in OPL.

   To select one of your own projects for copying and reuse, navigate to the folder that contains it and open it.

   Note: This procedure is only for OPL 6.x projects. For importing projects from previous releases of OPL, see the *Migrating from previous versions of OPL* section of the Migration Guide.

3. In OPL Projects Navigator, right-click on the project name and choose **Copy** from the context menu:

   You could also select the project name and press **Ctrl + C**.
4. Right-click again and choose **Paste** from the context menu, or press **Ctrl + V**.
5. A Copy Project popup window appears. Change the **Project name** to something other than the original project name or one of the other project names currently open in OPL, change the **Location** if necessary, and click **OK**.

   The new project appears in the OPL Projects Navigator.
6. Right-click the original project (the one you just copied) and choose **Delete** from the context menu. When the popup asks you if you want to delete all contents, choose **Do not delete contents**.

You can now modify the copied project as you want.

Note: You can use the **File>Copy Files to Project** menu command to open a dialog box that allows you to open files and import them into an existing project.

You can also drag existing files from a Windows Explorer window and drop them onto the project folder in your OPL Projects Navigator. This is always a *copy* operation, not a move.

Working with several projects

The IDE allows you to open more than one project at the same time, but only one of the open projects is active at once.

Setting the active project

The active project name is *the one that is currently selected in the OPL Projects Navigator*. No action is required to change active projects other than clicking the project name or any of the files inside the project folder.

Important: This does NOT mean that clicking in a project and making it the active project causes its default run configuration to be the active run configuration under the **Run** button  . By default, when you click the green arrow in the **Run** button, you relaunch the last run configuration solved, in whatever project was last run.

So if you want to launch a run configuration from the project you have just set as the active project by clicking in it, use the **Run** option from the right-click context menu, and select the run configuration you want. After the first time you do this, the run configuration will be set as the most recently run, and simply clicking the green arrow of the **Run** button will relaunch it.

Multiple references to the same file

There are two ways of reusing model, data, and settings files.

In several projects

You can create several projects that reference the same files. This makes sense, for instance, if your projects are meant to be eventually deployed in different business contexts while addressing similar or related optimization problems.

In several run configurations

If you plan to use OPL to write and test several versions of identical or similar optimization problems, you should rather consider creating several run configurations of a single project rather than one project for each variation. See *Executing a project* in *Getting Started with the IDE* for details. This can be the case, for instance, at prototyping stage, for testing and debugging purposes. Several samples delivered with the OPL distribution, such as `cutstock`, `production`, `sailco`, and others, define more than one run configuration.

Using templates

In IBM® ILOG® OPL, templates are regular `.mod`, `.dat`, and `.ops` files with a default content to help you start your own new files more quickly.

Using the default templates

When you use the File>New main menu item to create a new OPL model (`.mod`), data (`.dat`), or settings (`.ops`) file, predefined templates are inserted at the top of each file as a header. In OPL 6.x these templates are not user-modifiable or customizable.

Typically, a template contains placeholder strings which are automatically replaced by predefined values. For instance, the predefined `default.mod` template contains the following header:

```
/*********************************************
 * OPL %v Model
 * Author: %a
 * Creation Date: %d at %t
 *********************************************/
```

in which

This...	is replaced by...
%a	the Windows user name
%d	the date
%t	the time
%v	the version of OPL

Order of files

The order in which you add data files and settings files to a run configuration is significant.

More precisely, the position of the model in the run configuration and the position of data files relative to settings (`.ops`) files have no consequence. What may affect the results is the position of data files relative to each other and the position of `.ops` files relative to each other.

Order of data files

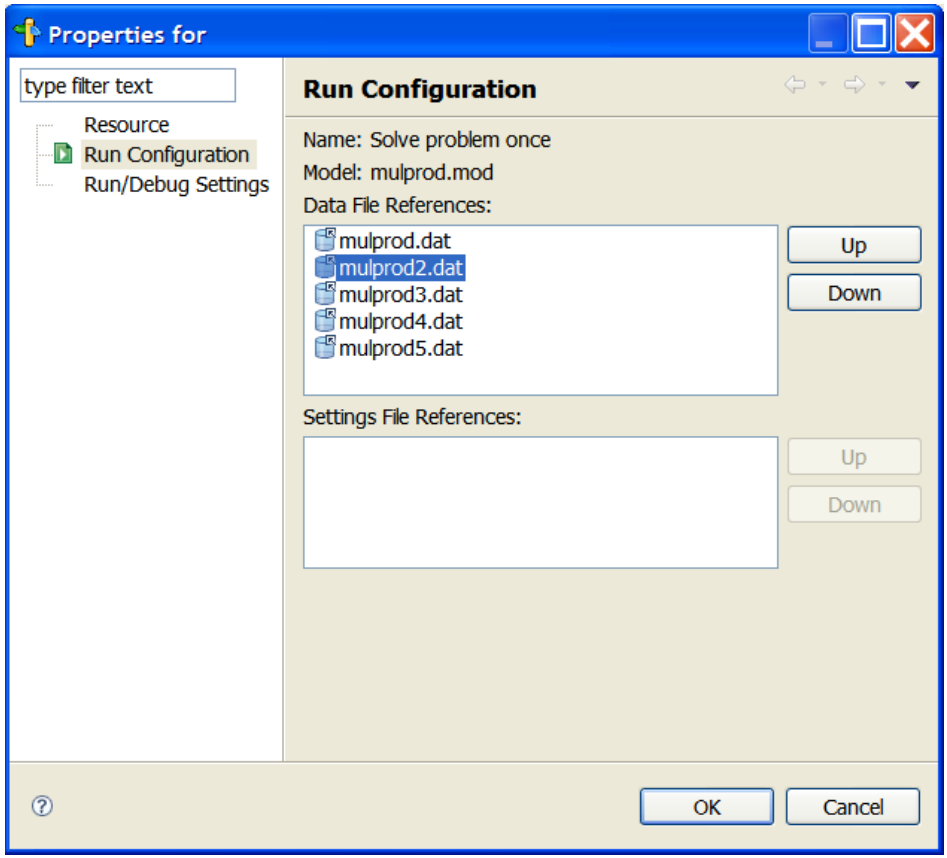
When you execute the project, the data files are called *in the order in which you added them to the run configuration*. Since some data in a `.dat` file may depend on other data in a different `.dat` file, adding data in the wrong order may cause an error at execution time.

Therefore, drag your data files into a run configuration in the order in which you want them to be executed.

To reorder data files within a run configuration:

1. Right-click on the run configuration name and choose **Properties** from the context menu.

   A properties window appears for the run configuration:



2. Use the **Up** and **Down** buttons to rearrange the order of your settings files and data files.

Order of settings files

If you add more than one `.ops` file to a run configuration, again the order in which you add them to the run configuration determines the order in which they will be executed. And again, you can change the order by right-clicking the run configuration and choosing **Properties**.

And as with data files, your results may be different depending in what order you add the `.ops` files. *The general rule is that for each setting, the non-default setting in the last settings file executed will "win."*

Example 1
For example, say that you add two settings files — `1.ops` and `2.ops` — to the same run configuration, and in that order. `1.ops` will execute first, followed by `2.ops`.

If in `1.ops` you leave the value of the CPLEX parameter **Algorithm for continuous problems** set to its default value, and in `2.ops` you change it to the **Dual simplex** value, the run configuration will be executed using the **Dual simplex** setting.

Example 2

Now imagine the same two settings files in the same run configuration, and in the same order.

As before, in `2.ops` you change the value of the **Algorithm for continuous problems** setting to **Dual simplex**. But this time, in `1.ops`, you change the value of *another* setting to something other than its default value. When this run configuration is executed, it will use both the **Dual simplex** setting found in `2.ops` **AND** the non-default value of the other setting found in `1.ops`.

Saving a project

To save your active project:

1. Choose **File>Save** from the main menu or press the keyboard shortcut **Ctrl+S**, or click the **Save** button  in the standard toolbar.

   The active project will be saved.

2. If nothing has been modified in the project, the **File>Save** menu option will not be active.

To save multiple projects:

1. Choose **File>Save All** from the main menu or press the keyboard shortcut **Ctrl+Shift+S**.

   All open projects in the OPL Projects Navigator that have changed information will be saved.

2. If nothing has been modified in any of the projects, the **File>Save** menu option will not be active.

Closing projects

Once you have saved the project, you can close it.

- Right-click on the project folder in OPL Projects Navigator and choose **Close** from the context menu.

When you close a project, it remains in the OPL Projects Navigator.

To remove it from OPL Projects Navigator, you must use the **Delete** context menu option.

Generating output files

Describes how the IDE enables you to generate model and data as output files that you can later use from a command prompt and with OPL interface libraries.

In this section

Generating a compiled model
Describes how to generate a binary version of a model file.

Exporting external data
Describes how to export externally-initialized data.

Exporting internal data
Describes how to export internally-initialized data (calculated data).

Generating a compiled model

From the IDE, you can generate a compiled model. A compiled model is a binary version of a `.mod` file.

Keeping your models as compiled files may help you protect your intellectual property because these binary files are not human-readable. You can also pass a compiled model file name as an argument to a method of the OPL interface libraries. See the *Interfaces User's Manual* for more information.

From version 5.1 of OPL onwards, compiled model files are detected automatically by **oplrun**, interfaces related to `IloOplModelSource`, interfaces related to `IloOplRunConfiguration`, and scripting statements.

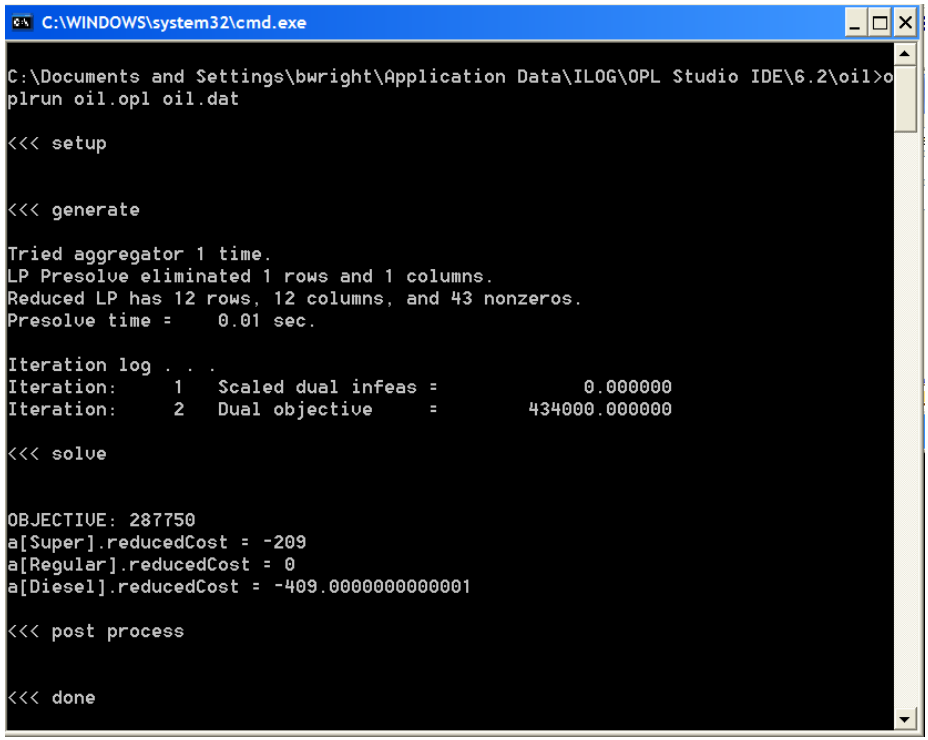
To generate a compiled model:

1. Open a project in the usual way and select the model you want to create a compiled model for.
2. Right-click the model and choose **Compile model** from the context menu. See *Model, data, and settings files* in the topic *Right-click context menu commands*.

   The compiled model file is created and added to the folder that contains the model file, with the same name as the model and a file extension of `.opl`.
3. To run the compiled model, open a command prompt (on Windows platforms) and enter the `oplrun` command in your workspace. For example:

   ```
   C:\Documents and Settings\<your_name>\Application Data\ILOG\OPL Studio
   IDE\<version>\gas\oplrun gas.opl gas.dat
   ```

   The following result is displayed in the MS DOS window.



Note: You cannot load compiled models (`.opl` files) into the IDE.

You can also use the `oplrun` command on a supported UNIX platform. See the oplrun Command Line Interface document.

Exporting external data

The IDE also enables you to export external data in a model to an output file (a `.dat` file in most cases). All data elements in a `.mod` file declared by means of the ellipsis syntax

```
= ...;
```

are external. For example, `Products`, `Components`, `Demand`, `Profit`, and `Stock` are external data in the following sample (`gas.mod`).

External data in a model (`gas` project)

```
{string} Products = ...;
{string} Components = ...;

float Demand[Products][Components] = ...;
float Profit[Products] = ...;
float Stock[Components] = ...;
```

Since these data elements are resolved from data sources (such as databases, spreadsheets) when the OPL model is instantiated, exporting external data amounts to dumping the data from such data sources to a local `.dat` file.

To export external data:

1. Open a project in the usual way and execute the run configuration you want to export data for.
2. When the run has completed, open the Run menu in the main menu bar and choose **Export external data**.

   A dialog box appears that allows you to choose the filename and location of the saved file. The default extension of the generated file is `.dat`, and by default, the file is saved in the same directory as the model file.
3. Type a name for the file (for example, `gasext.dat` to avoid overwriting the distributed `gas.dat` file), and click **Finish**.
4. Check for the resulting file (for example, in Windows Explorer) in the location you specified.

   The generated data file looks like a regular data file and you can open it in a text editor or in the IDE (although it does not appear in the project tree).

> **Note**: If the data is not used in the model, it will not be included in the generated data file.

Exporting internal data

The IDE also enables you to export internal data in a model to a local `.dat` file. This may be useful for debugging purposes. Internal data are all data elements in a `.mod` file initialized by means of the syntax

```
= "(some expression here)"
```

For example, `Routes`, `Supplies`, and `Customers` are internal data in the following sample: (`transp4.mod`).

Internal data in a model (`transp4.mod`)

```
{route} Routes = { < p,<o,d> > | <p,o,d,c> in TableRoutes };
{connection} Connections = { c | <p,c> in Routes };
tuple supply{
  string p;
  string o;
}
{supply} Supplies = { <p,c.o> | <p,c> in Routes };
float Supply[Supplies] = ...;
tuple customer {
  string p;
  string d;
}
{customer} Customers = { <p,c.d> | <p,c> in Routes };
float Demand[Customers] = ...;

float Cost[Routes] = [ <t.p,<t.o,t.d>>:t.cost | t in TableRoutes ];
```

To export internal data from a model:

1. Open a project in the usual way and execute the run configuration you want to export data for.
2. When the run has completed, open the Run menu in the main menu bar and choose **Export internal data**.

   A dialog box appears that allows you to choose the filename and location of the saved file. The default extension of the generated file is `.dat`, and by default, the file is saved in the same directory as the model file.
3. Type a name for the file (for example, `gasint.dat` to avoid overwriting the distributed `gas.dat` file), and click **Finish**.
4. Check for the resulting file (for example, in Windows Explorer) in the location you specified.

   The generated data file looks like a regular data file and you can open it in a text editor or in the IDE (although it does not appear in the project tree).

Note: If the data is not used in the model, it will not be included in the generated data file.

Launching OPL run configurations in the background

Explains the two methods of launching OPL run configurations from the IDE as a background process, and the options available when doing this.

In this section

Running your projects in the background
Describes how to use the External Tools button in the execution toolbar to launch your OPL run configurations from the IDE as background processes.

Creating an oplrun launch configuration interactively
Describes the first method of launching OPL run configurations as background processes.

Creating a customized oplrun launch configuration
Describes the second method of launching OPL run configurations as background processes and saving them as predefined launch configurations.

External Tools Dialog options
This section explains the options available to you on the four tabs of the External Tools Dialog.

Running your projects in the background

As described in the *The Run options* section of *Getting Started* and in *The execution toolbar* section of this manual, you can use the **Run**, **Debug**, and **Browse** buttons of the execution toolbar to run your OPL projects interactively within the OPL IDE.

You can also use the **External Tools** button to set up *launch configurations* that allow you to run your OPL projects in the *background*, using `oplrun`.

This allows you to create launch configurations for different run configurations in your projects. You can then launch them as background processes from the OPL IDE, and monitor their progress in the Console tab of the Output Area.

Two ways to launch an OPL run configuration from the IDE

Using the **External Tools** button on the execution toolbar, there are two basic methods of launching a run configuration in the background:

- **Interactively** — select the run configuration you want to run in the background in OPL Projects Navigator, and then select the **oplrun** item in the External Tools Dialog to run it.
- **Create predefined launch configurations** — duplicate the **oplrun** item in the External Tools Dialog, rename it to specify the run configuration, and then use the options in the dialog to custom-configure

Details on these procedures are contained in the two following sections.

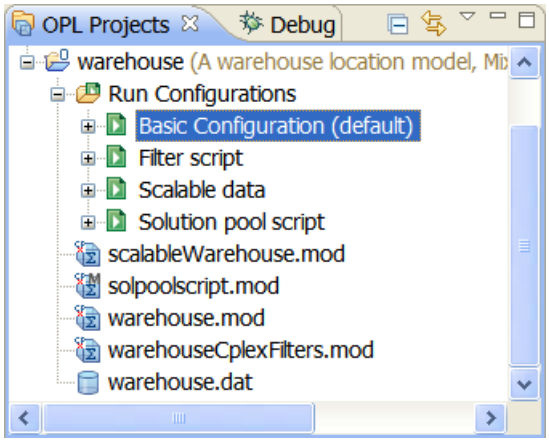
Creating an oplrun launch configuration interactively

The simplest method of launching an OPL run configuration in the background is to select that run configuration in OPL Projects Navigator, and then select the **oplrun** item in the External Tools Dialog and click **Run** on that dialog. Predefined variables handle the project's **Location**, **Working Directory**, and standard **Arguments**.

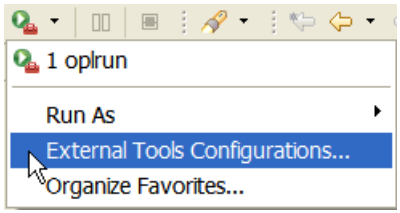
To launch an OPL run configuration as a background process interactively:

1. In OPL Projects Navigator, open the project and select the run configuration you want to run.

   In this procedure, we will use the `warehouse` example from the OPL distribution, and select the **Basic Configuration (default)** run configuration.



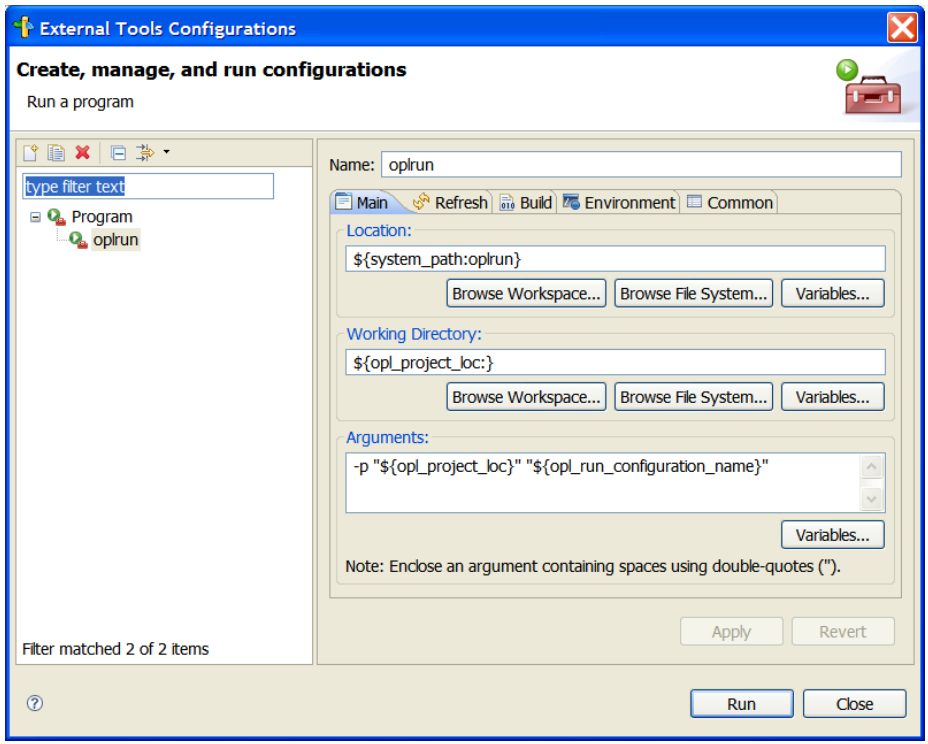
2. In the execution toolbar, click the arrow to the right of the **External Tools** button to display its contents.



> **Note**: If you have never used this item before, the numbered **1 oplrun** launch configuration entry at the top will not be visible.

3. Select **Open External Tools Dialog**.

The dialog is displayed. If the default **oplrun** is not highlighted in the left pane, select it so that your window looks like the following screenshot:



Note that the three input fields are already filled in with variables and/or arguments:

- **Location** — this field contains the default variable `${system_path:oplrun}`.

  This instructs OPL to use the project selected in OPL Projects Navigator as the project location.

- **Working Directory** — this field contains the default variable `${opl_project_loc:}`.

  This instructs OPL to use the standard working directory for the project selected in OPL Projects Navigator.

- **Arguments** — this field contains the default argument and variable `-p "${opl_project_loc}" "${opl_run_configuration_name}"`.

  The `-p` flag instructs OPL to execute the run configuration at the path that follows it. In this case, that is the run configuration selected in OPL Projects Navigator.

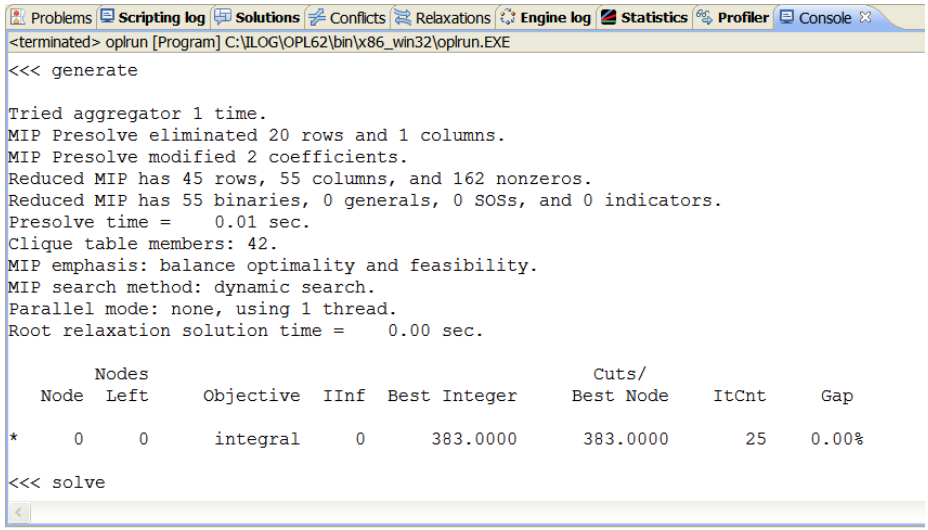
You can run the model at this point.

Note: You could type in additional `oplrun` arguments in the **Arguments** field, but this is not recommended because the **oplrun** entry is a *template* that can be duplicated and used to create and save more complex launch configurations.

If you add the arguments to the **oplrun** template, they will be saved and will be present in the future whenever you duplicate it, using the instructions given in the *Creating a customized oplrun launch configuration* section that follows this procedure.

4. Click the **Run** button on the External Tools Dialog to launch the run configuration using `oplrun`.

   In the Output Area, note that a new **Console** tab appears. This is where you can use the icons in its toolbar to control the execution and display of the background process (as explained in *The Output Area tab views toolbars*). You can also view the output from the background process as it appears.



This is the simplest method of launching an OPL run configuration as a background process using `oplrun`. Its advantage is that it allows you to solve long-running models in the background and monitor their progress in the **Console** tab, while continuing to work in the OPL IDE.

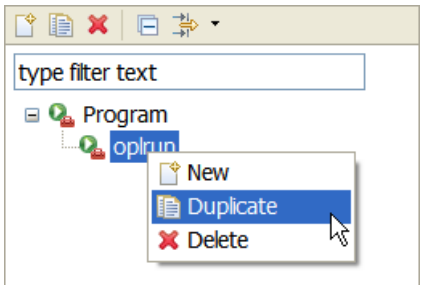
Creating a customized oplrun launch configuration

Another method of launching OPL run configurations in the background, especially if the runs require additional `oplrun` command line arguments, environment variables, or changes to standard input and output, is to define and save launch configurations.

This is done using the same External Tools Dialog, by duplicating the **oplrun** template, renaming it, and specifying the exact arguments and other variables you want to use for the run.

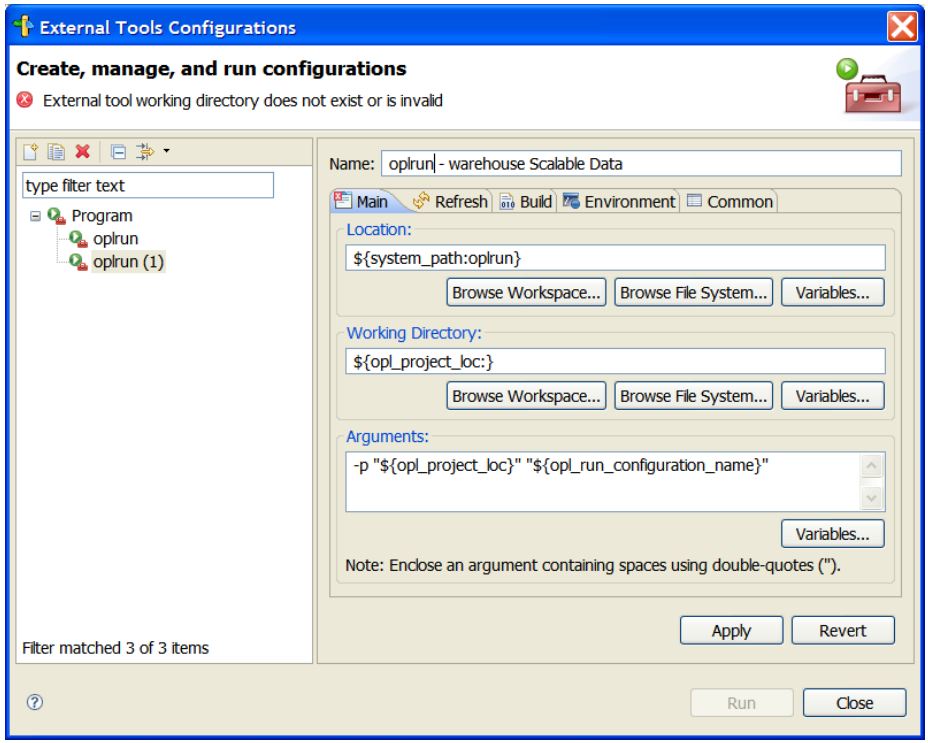
To create and save a customized `oplrun` launch configuration:

1. In the execution toolbar, click the arrow to the right of the **External Tools** button and select **Open External Tools Dialog** to display the dialog.
2. In the External Tools Dialog, right-click the **oplrun** template item and select **Duplicate** from the context menu.



3. A duplicate of the **oplrun** template is created with the default name **oplrun (1)** and appears in the left pane of the dialog.

   Your first step at this point should be to rename the duplicate and give it a meaningful name, as has been done in the screenshot below:



The following sections contain explanations of the options available to you on the four tabs of the External Tools Dialog.

To create your customized launch configuration, just select or type in the options you want, and click **Apply**, which saves the changes. (You can revert to the condition of your last save by clicking the **Revert** button.)

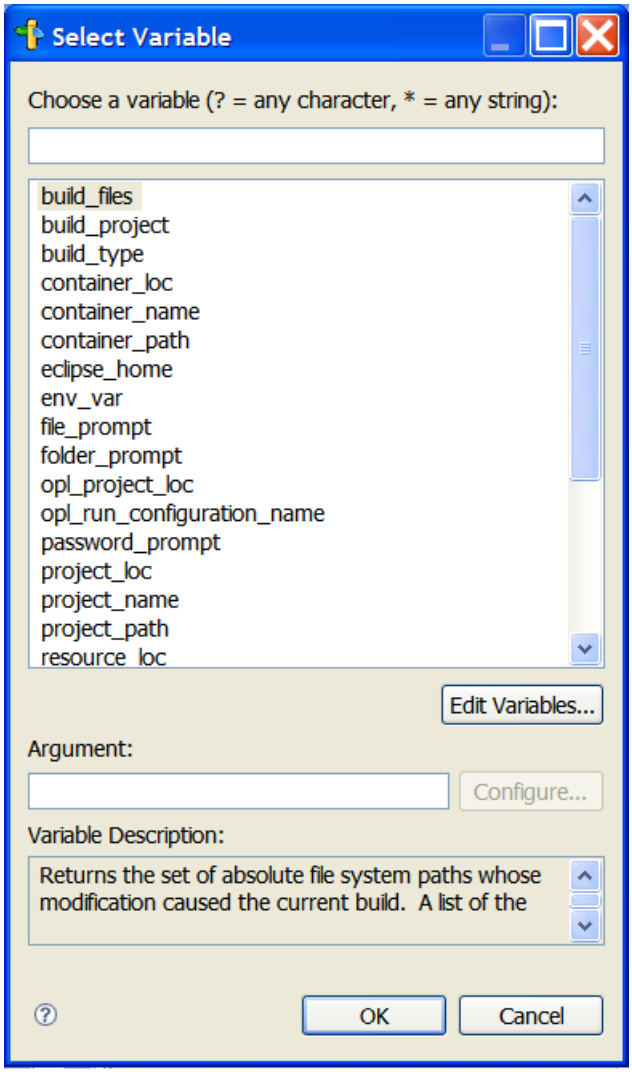
When you have finished customizing your launch configuration, click the **Run** button to launch it in the **Console** tab of the Output Area. The launch configuration is then saved and available to use again at any time in the future.

External Tools Dialog options

The Main tab

On the **Main** tab of the External Tools Dialog, you specify or customize the contents of the **Location**, **Working Directory**, and **Arguments** fields.

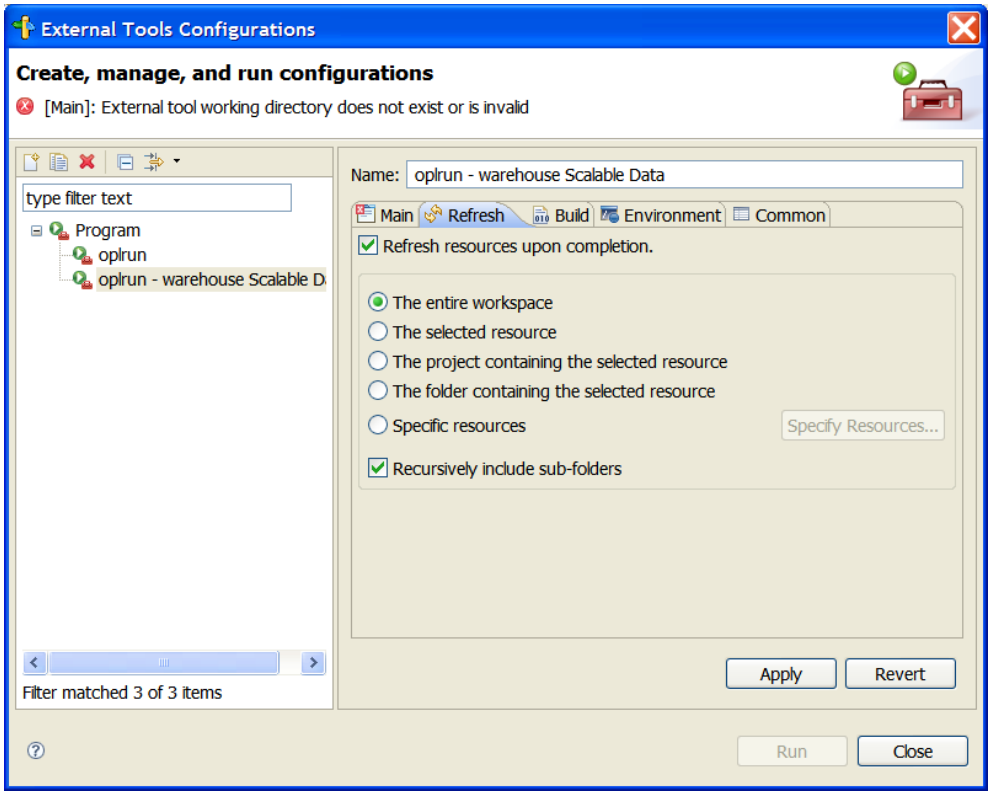
- In the **Location** and **Working Directory** fields, you can use the **Browse Workspace** and **Browse File System** buttons to search for the project you want to define a launch configuration for.
- On all three tabs, you can click the **Variables** button to display a popup list of variables that can be inserted into the fields. (To determine what the variables are, click on one in the list and read its description at the bottom of the popup window.)



- In the **Arguments** field, you can type in any standard `oplrun` command line arguments.
- In the **Arguments** tab, you can also enter the name of the run configuration you want to run, inside double quotes.

The Refresh tab

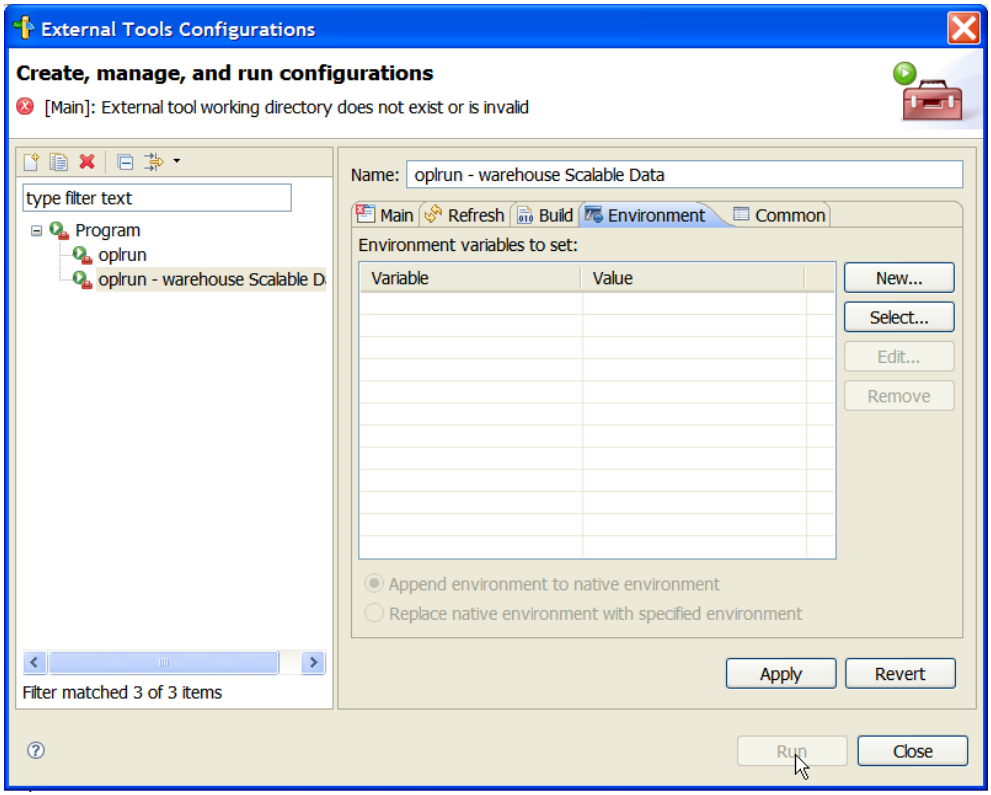
On the **Refresh** tab, you choose which project resources should be refreshed at the completion of the run.



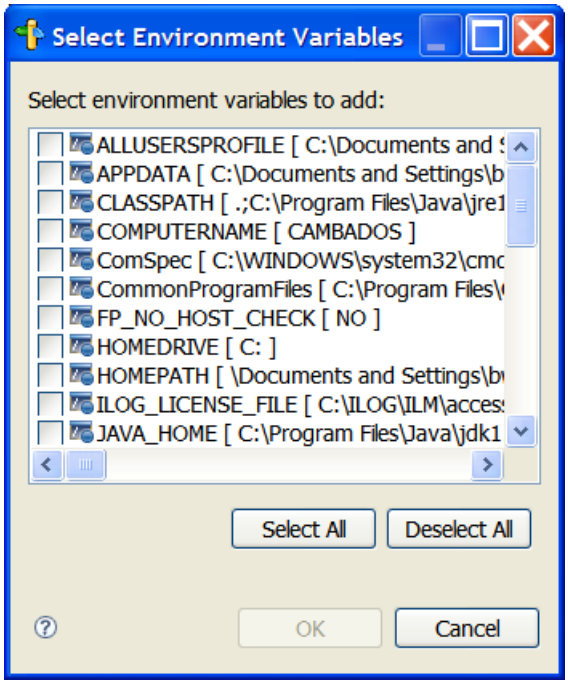
- To specify, check the **Refresh resources upon completion** box, and then select from the list below.
- Check **Recursively include sub-folders** if you want to Refresh the entire hierarchy selected.

The Environment tab

On the **Environment** tab, you can choose which variables you want to inherit from the Windows environment for this run, and you can also specify additional environment variables.



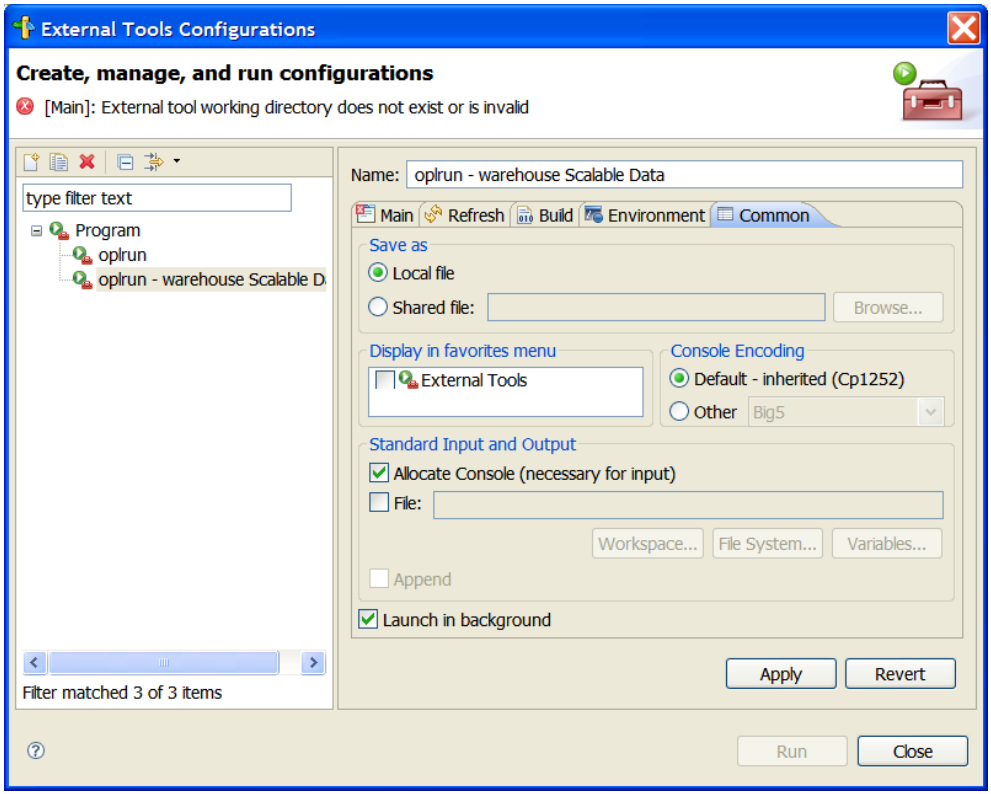
- Click **New** to specify new environment variables for this process.
- Click **Select** to display a popup window of all variables inherited from the Windows environment.



On this popup window, you can choose to inherit all of the Windows environment variables (the default, if you do not hand-select variables on this popup window), or select only the variables you want to use when this process is run.

The Common tab

On the **Common** tab, you can specify a Save As location for the launch configuration, choose to display the launch configuration in your Favorites menu, control the Console Encoding, and specify an optional location for the standard input and output files.



- In the **Save As** area, you can specify a location for the launch configuration in the **Shared As** field. For example, you could save it to a network drive so that all of the members of your OPL development team could use the same launch configuration.
- In the **Standard Input and Output** area, you can use the **Workspace**, **File System**, and **Variables** button to specify a location for optional input and output files.

IDE Preferences

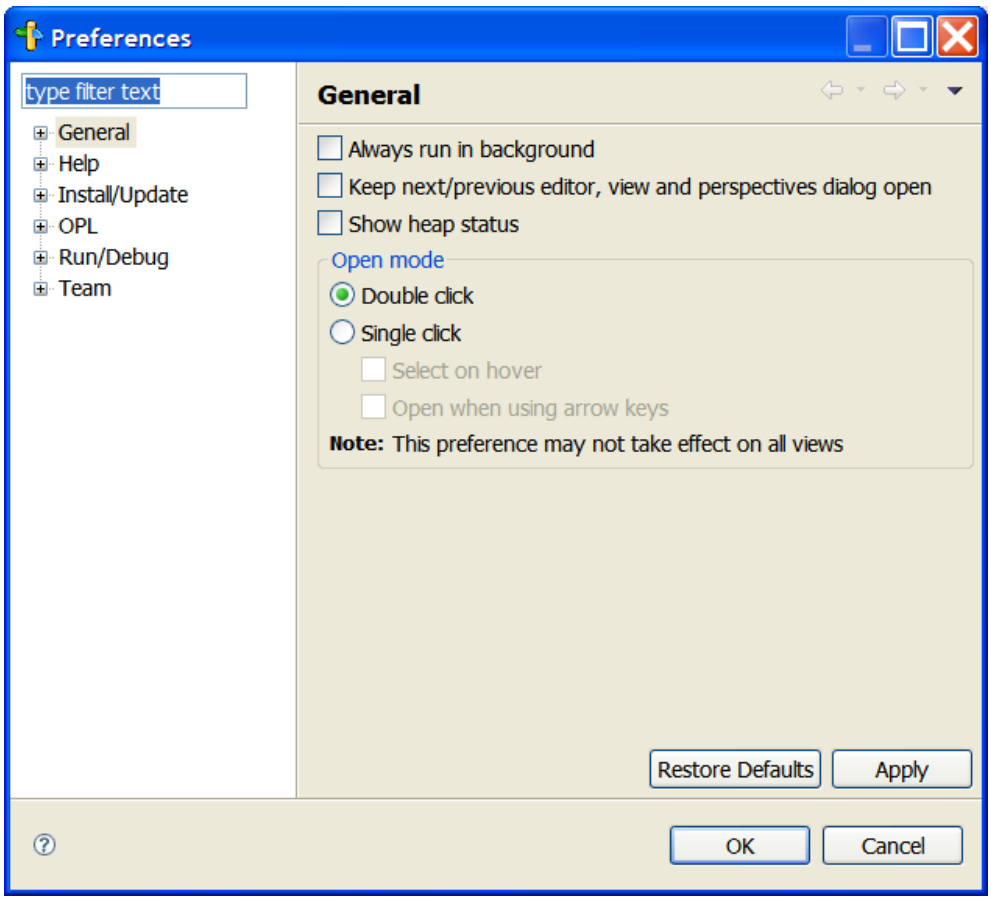
Describes how to customize the appearance and features of the graphical user interface and the behavior of the editor and other tools such as CVS by setting preferences for the IDE.

In this section

The Preferences dialog box
Describes how to access the IDE preferences and what kind of preferences you can change.

The Preferences dialog box

Choose **Window>Preferences** in the main menu to open the Preferences dialog box:



This dialog enables you to change the following settings:

- **General**
  - Appearance
    - Colors and Fonts
    - Label Decorations
  - Compare/Patch
  - Content Types
  - Editors

- File Associations
- Text Editors
  - Accessibility
  - Annotations
  - Hyperlinking
  - Linked Mode
  - Quick Diff
  - Spelling
- Keys
- Perspectives
- Startup and Shutdown
- Welcome
- Workspace
  - Build Order
  - Linked Resources
  - Local History

- **Help**
  - Content
- **Install/Update**
- **OPL**
  - General Preferences
    - Data file size limit (expressed in bytes)
  - Colors
- **Run/Debug**
  - Console
  - External Tools
  - Launching
    - Default Launchers
    - Launch Configurations
  - Perspectives

- String Substitution
- View Management

- **Team**
  - CVS
    - Annotate
    - Comment Templates
    - Console
    - Ext Connection Method
    - Label Decorations
    - Password Management
    - Synchronize/Compare
    - Update/Merge
    - Watch/Edit
  - File Content
  - Ignored Resources
  - Models

How to use the Preferences dialog

Setting preferences in the OPL IDE is straightforward and does not require step-by-step instructions.

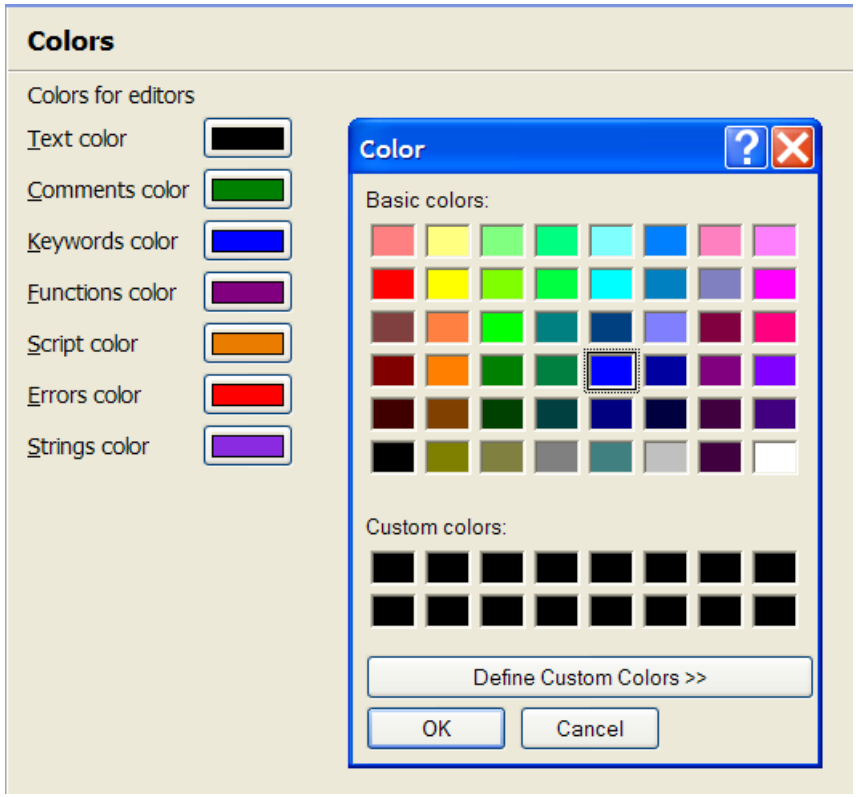
To change settings:

1. In the left pane of the Preferences window, click the category of setting you want to change.

   Detailed setting options for that category appear in the right pane.
2. Change the settings as desired and click **Apply**.
3. To revert a changed setting back to its default value, click **Restore Defaults**.
4. Close the Preferences dialog when you are finished by clicking **OK**.

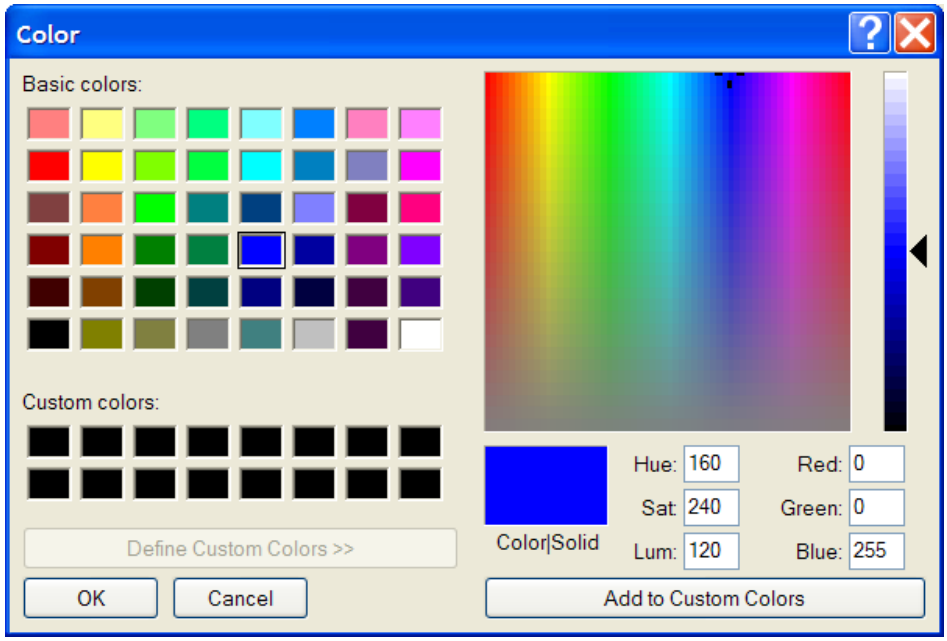
To change color settings:

1. In a preference editor that allows you to choose colors, to change them click one of the color boxes to display the Color dialog:



Detailed setting options for that category appear in the right pane.

2. Choose a color from the list and click **OK**.
3. If none of the colors suit you, click **Define Custom Colors** and use the additional features to add your own colors to the list.



4. Close the Color dialog when you are finished by clicking **OK**.

OPL IDE advanced settings

The OPL IDE accepts the following settings from the command line:

- `oplide -vmargs`, to customize the operation of the Java VM used to run the OPL IDE. If specified, this option must come at the end of the command line.

  For example, `oplide -vmargs -Xmx512m` runs the OPL IDE with a Java VM of size 512 MB.

- `oplide -consoleLog`, to obtain a log of events in the console.

- `-nl <locale>`, to change the locale.

  For example, `-nl ja_JP`, to obtain an IDE in Japanese.

- `-clean`, to clear the cache.

- `-data <dir>`, to use a different workspace, where <dir> represents the directory name of the new workspace.

- `-oplOpenFile`, to open a file in an existing IDE or create a new IDE instance.

- `-noSplash`, to hide the splash screen.

The following settings are for very advanced use:

- `-vm <path to Java VM>`, to locate the Java VM to run the OPL IDE.

  The path should be the full file system path to an appropriate `Java jre/bin` directory, Java executable, Java shared library (`jvm.dll` or `libjvm.so`), or a Java VM Execution Environment description file.

- `-debug`, to debug in verbose mode.

- `-console`, to open an OSGi console for advanced debugging.

OPL IDE memory allocation

You can change the amount of memory available to the OPL IDE in the file `<OPL_dir>\oplide\oplide.ini`.

It contains the following settings by default:

```
--launcher.XXMaxPermSize
128m
-vmargs
-Xmx1024m
-Xms40m
```

`--launcher.XXMaxPermSize`	The size of the permanent generation heap allocated to the IDE launcher. The default size is 128 MB.
`-vmargs`	Arguments passed directly to the Java virtual machine.
`-Xmx1024m`	The default maximum memory allocation to the virtual machine is 1 GB.
`-Xms40m`	The default initial memory allocation to the virtual machine is 40 MB.

Note: If you are changing the `oplide.ini` file, please keep the following restrictions in mind:

- If you are adding additional flags to the file, and still want to use the `-vmargs` flag to modify maximum or initial memory allocation, the `-vmargs` flag and its arguments must be the last entries in the file, as they are above.
- The `-vmargs` flag and its two arguments should each be on their own line in the `oplide.ini` file. These individual lines are concatenated at runtime, so the memory management request in the default `oplide.ini` file shown above would be expressed on the command line as:

  ```
  oplide -vmargs -Xmx1024m -Xms40m
  ```

- Do not concatenate these arguments onto one line in the `oplide.ini` file. This will result in the flag being ignored.

Index

Symbols

! (key fields) tuple syntax
  in OPL error messages **26**
  in the OPL IDE Outline **26**

A

Abort button **43**
Abort Execution button **47**
About, Help menu command **98**
accessing
  data files **140**
active project **130**
advanced settings **163**
arrays
  icon types in Problem browser **108**
available memory in the IDE **164**

B

Breakpoints view
  toolbar **47**
Breakpoints view toolbar
  toolbar **47**
Browse As command **94**
Browse button **28**
Browse Configurations command **94**
Browse History command **94**
buttons
  Browse **28**
  close current document **89**
  Cut **91**
  Debug **94**
  description **87**
  in document toolbar **50**
  in project toolbar **51, 52**
  in standard toolbar **41**
  Run **94**
  Save **89**
  Save All **89**
  tooltips **39**

C

call stack
  icons for types and sets **109**
check mark
  in Problems tab **102**
Clear contents, right-click command in Output Area tabs **102**
clipboard
  copying contents of Output Area tab into **102**
Close All command **89**
Close button **89**
closed project **18**
closing
  a project **136**
  the active file **89**
column number for cursor position, in Status Bar **35**
command line
  settings available from **163**
Comment/Uncomment command **91**
Compare With **81**
Compile selected model, menu command **138**
compiled model files
  generating and using **138**
constants
  icon types in Problem browser **108**
constraints
  expanding **122**
  icon types in Problem browser **108**
  in Problem browser, labels **28**
context menus **99**
Continue button **47**
Copy command **91**
Copy contents to clipboard, right-click command
  in Output Area tabs **102**
Copy Files to Project command **89**

copying OPL projects **21, 128**
Cut command/button **91**

D

data
external, exporting **140**
icon types in Problem browser **108**
internal, exporting **141**
table view in Problem Browser **126**
data files
order **132**
referenced by several projects **130**
right-click commands **100**
template **131**
data sources
dumping to a local data file **140**
Debug As command **94**
Debug button **43, 47**
Debug command **94**
Debug History command **94**
Debug view
toolbar **47**
Debug view toolbar
toolbar **47**
decision variables
icon types in Problem browser **108**
table view in Problem Browser **126**
default .mod .dat, and .ops templates **131**
default values
of IDE preferences **104**
restore **104**
Delete command **91**
displaying
results **126**
drag and drop a file to the IDE **128**
Dynamic Help **98**

E

Editing Area
description **25**
Editor
how to use **55**
quick reference **60**
resizing window **59**
switching between windows **58**
ellipsis
to declare data **140**
Engine log output tab
saving the log to a file **37**
error messages
! (key fields) syntax in **26**
! (key) symbol **26**
key fields (!) syntax in **26**
executing
Debug button **43, 47**
Debug command **94**
Run command **94**
execution
displaying results **113**
events, description **112**
pausing and resuming **43, 47**
toolbar **43, 47**
Execution menu
equivalent buttons **43, 47, 144, 145, 148**
Exit command **89**
expanding constraints **122**
Export external data command **94**
Export External Data menu command **140, 141**
Export internal data command **94**
External Tools button **43, 144, 145, 148**
launching oplrun from the OPL IDE **43, 144, 145, 148**
oplrun **43, 144, 145, 148**
oplrun launch configurations **43, 144, 145, 148**
External Tools command **94**
External Tools Dialog
Common tab **150**
Environment tab **150**
Main tab **150**
Refresh tab **150**

F

F1 key, contextual help **25, 88**
File menu **89**
equivalent buttons
for projects **41**
File Search feature **23**
file types
file name extensions, description **22**
files
opening **128**
order in a project **132**
referenced by several projects **130**
renaming in the IDE **101**
settings.xml **104**
Find Next command **91**
Find Previous command **91**
Find/Replace command **91**
forall, statements
details **122**
Force element usage, OPL Language option **120**

G

Go To Line command **93**

H

Help menu **98**

I

icons
for types in Problem browser **28, 108**
in a script-debugging call stack **109**
IloOplModelSource class

automatic detection of compiled model files **138**
IloOplRunConfiguration class
automatic detection of compiled model files **138**
Import
Existing OPL 6.x projects **89**
OPL-ODM 5.x projects **89**
Import command **89**
Incremental Find Next command **91**
Incremental Find Previous command **91**
indented blocks **57**
integer solutions
notification **113**
interface libraries **138**
internal data
exporting and using **141**

K

Key Assist **98**
key F1 **25, 88**
key fields (!) tuple syntax
in OPL error messages **26**
in the OPL IDE Outline **26**
keyboard shortcuts in editor **60**
keyword help **25, 88**
keywords **25**
color-coded in IDE Editing Area **25**

L

labeled constraints
in Problem browser **28**
language keywords
color-coded in IDE Editing Area **57**
launch configurations **144, 145, 148**
resuming execution **43**
launching oplrun from the IDE **144, 145, 148**
line numbers
for cursor position, in Status Bar **35**
Local History **81**
log files **37**

M

main window
description **11**
Editing Area **25**
line number, column number **35**
menu commands **87**
name of current file **35**
Outline **26**
Output Area **37**
Problem browser **28**
project tree **18**
Status Bar **35**
MDI (Multi-Document Interface) **57**
memory allocation and management
in the OPL IDE **164**
menu bar
commands and equivalent toolbar buttons **87**
menu commands
Compile selected model **138**
Edit menu **91**
Export External Data **140, 141**
File menu **89**
Help menu **98**
Navigate menu **93**
Run menu **94**
Window menu **97**
migrating projects to OPL **89**
Model Completion
Ctrl+space syntax to invoke **63, 65**
Dot (.) syntax to invoke **63, 65**
in the OPL IDE editor **63, 65**
Model Templates
creating **76**
editing **76**
in the OPL IDE editor **63, 73, 76**
models
Outline **26**
right-click commands **100**
template **131**
multiple levels of undo and redo **57**

N

New command **89**

O

ODM Documentation **98**
online help
accessing **25**
Open Debug Dialog command **94**
Open File in Editor command **89**
open project **18**
Open Run Dialog command **94**
opening files **128**
opening projects in OPL **89**
OPL Documentation **98**
OPL IDE editor
Model Completion **63, 65**
Model Templates **63, 73, 76**
OPL Keyword Help **98**
OPL Preferences editor
creating Model Templates **76**
editing Model Templates **76**
OPL Projects Navigator
description **18**
OPL Projects Navigator toolbar
description **50**
oplrun
creating custom launch configurations **144, 145, 148**
launching from the IDE **144, 145, 148**
optimal solution **113**

options
setting in IDE **103**
orange status indicator **112**
order
in Problem browser **28**
in the Outline **26**
of data files **132**
of settings files **133**
Outline window
! (key) symbol **26**
description **26**
Output Area
Conflicts tab **37**
Console tab **37, 43**
Console view **144, 145, 148**
description **11, 37**
Engine log tab **37**
Problems tab **37**
Profiler tab **37**
Relaxations tab **37**
right-click commands **102**
Script log tab **37**
Solutions tab **37**
Statistics tab **37**
Output Area tabs toolbar
description **52**
output files, generating **137**

P

Paste command **91**
Pause button **43**
Pause Execution button **47**
postprocessing
in Problem Browser **124**
preferences
setting IDE preferences **103**
Preferences command **97**
Problem Browser
displaying problem results **126**
navigating the model file **122**
postprocessing **124**
Problem browser
description **28**
icons for types **108**
order of elements **28**
solution pool **28**
production example **117**
Progress chart **113**
project
closed **18**
open **18**
project toolbar
description **51**
project tree
description **18**
projects
copying in OPL Projects Navigator **21, 128**
file name extensions **22**
reusing **128**
right-click commands **100**
saving and closing **135**
setting as active **130**
Property table, in Problem browser **28**

R

ranges
icon types in Problem browser **108**
red exclamation mark/red star, default value changed **104**
Redo command **91**
renaming files **101**
Replace With **81**
resizing editor windows **59**
restoring default values **104**
results
displaying **113**
Resume command **94**
resuming execution **43, 47**
reusing existing OPL projects **128**
right-click commands **99**
Run As command **94**
Run button **43, 47**
functions and features **43**
Run command **94**
run configuration
launching as a background process **144, 145, 148**
run configurations
resuming execution **43, 47**
right-click commands **101**
Run History command **94**

S

Save All button **89**
Save All command **89**
Save As command **89**
Save command **89**
saving
projects **135**
several files **89**
search
in a file **23**
separators
within main window **11**
settings available from the command line **163**
settings files
order **133**
right-click commands **100**
template **131**
Settings Outline window **26**
settings.xml file **104**
shortcuts

Ctrl+Tab to change editor window **58**
to menu commands **87**
Show View command **97**
solution pool **28**
solutions
notification **113**
optimal **113**
standard toolbar
description **41**
states
running **113**
Status Bar **35**
run indicator **35**
see run status **35**
show background operations button **35**
Status Bar description **35**
status indicator
orange **112**
Status Bar **113**
status messages
during execution **112, 113**
strings
icon types in Problem browser **108**
syntax
ellipsis **140**

T

table views of model elements **125**
templates for model, data, and settings files **131**
text editor
how to use **55**
quick reference **60**
toolbars
Breakpoints view **47**
buttons **87**
Debug view **47**
description **39**
execution **43**
Variables view **47**
tooltips
for toolbar buttons **39**
tuple keys **26**
tuples
! (key fields) syntax in OPL error messages **26**
! (key fields) syntax in the OPL IDE Outline **26**
key fields (!) syntax in OPL error messages **26**
key fields (!) syntax in the OPL IDE Outline **26**

U

Undo command **91**
undoing/redoing **57**

V

Variables view
toolbar **47**
Variables view toolbar
toolbar **47**
variables. See decision variables **126**
Version list **81**

W

Welcome window command **98**
Window menu **58**
windows
resizing editor window **59**
switching editor windows **58**
Word Completion command **91**
Working Sets command **97**